Research report 101

THE BLACKBOARD ARCHITECTURE: EXAMPLE SYSTEMS

Iain D Craig

(RR101)

Abstract

This paper describes the principal features of some of the major implementations of the blackboard architecture. The architecture is becoming increasingly popular, yet most descriptions are to be found in conference papers and are of a somewhat fragmentary nature. This paper attempts to fill the need for a single collection of system descriptions. The descriptions are motivated by a general introduction to the blackboard model of problem solving.

Department of Computer Science
University of Warwick
Coventry
CV4 7AL, UK

May 1987

THE BLACKBOARD ARCHITECTURE: EXAMPLE SYSTEMS

Iain D. Craig

Department of Computer Science,

University of Warwick,

Coventry CV4 7AL, UK

ABSTRACT

This paper describes the principal features of some of the major implementations of the blackboard architecture. The architecture is becoming increasingly popular, yet most descriptions are to be found in conference papers and are of a somewhat fragmentary nature. This paper attempts to fill the need for a single collection of system descriptions. The descriptions are motivated by a general introduction to the blackboard model of problem solving.

May 6, 1987

THE BLACKBOARD ARCHITECTURE: EXAMPLE SYSTEMS

Iain D. Craig

Department of Computer Science,

University of Warwick,

Coventry CV4 7AL, UK

1. INTRODUCTION

The blackboard architecture for problem solving is becoming increasingly popular, particularly for systems which operate in real time contexts or in complex domains. The architecture provides a framework and a methodolgy within which to construct problem solving systems for such domains. The architecture is well-understood by those who have examined the comparatively scarse literature, but appears to be inaccessible to those who have not performed extensive bibliographic searches.

This paper presents the basic metaphor behind the architecture and examines some of the major examples of implemented blackboard systems in comparative detail. It is structured as follows. In section two, the basic model is presented in informal terms. The model has as its core the metaphor of a group of experts collaborating in the solution of a difficult problem. Section three is the kernel of the paper: it describes some of the major systems which have been implemented to date. The system descriptions are not intended to be complete: rather, they are intended to show what the contribution of each system to the development of the architecture was. As a consequence, the HEARSAY-II speech understanding system is presented first. HEARSAY-II was the first blackboard system and serves as a starting point for the development of the architecture. As Nii notes ([Nii, 1986a], [Nii, 1986b]), the architecture has developed as a rationalisation of the original HEARSAY-II system and each implemented system has added or removed

some features. The final section of the paper acts as a summary.

This present paper is intended to complement Nii's survey papers ([Nii, 1986a], [Nii, 1986b]). It also provides details of systems which she treats only briefly.

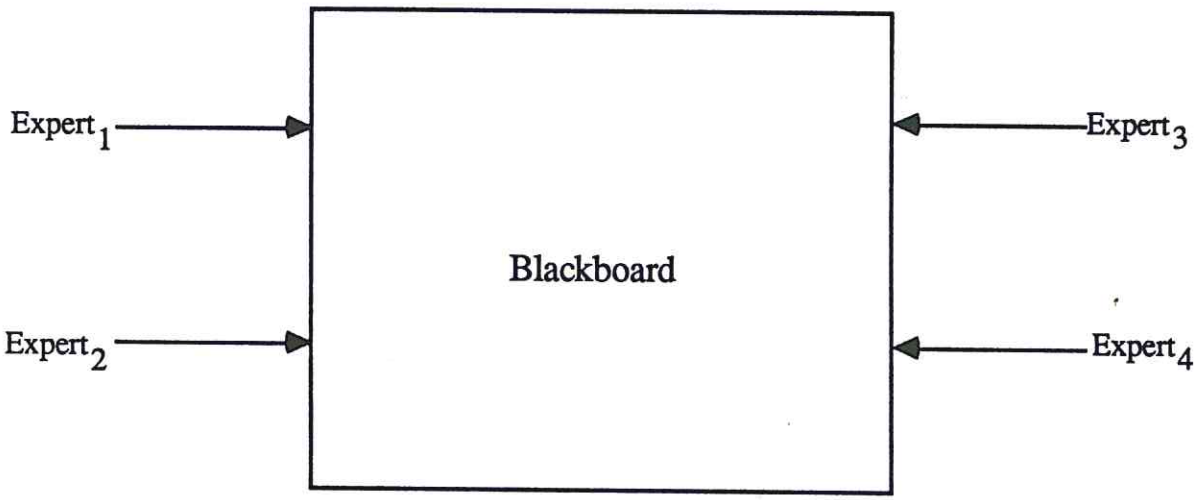
2. THE MODEL

Despite the fact that there have been many implementations of the blackboard model, they have all differed from one another in some respects, or else have been copies of HEARSAY-II (see section 3.2 below). What all the various implementations have in common is the fact that they have the same common model as a basis for their architectures. This point is important and has been noted by Barbara Hayes-Roth ([Hayes-Roth, 1983]). In this section, I will present the model in terms of the metaphor of experts and a traditional physical blackboard. Having presented the metaphor, I will discuss some extensions and implications of the metaphor, again in metaphorical terms.

2.1. The Metaphor

Imagine a group of experts trying to solve a complex problem. The experts are competent in different domains which are relevant to the problem. They stand around a blackboard. They communicate ideas for the solution of the problem by word of mouth, by diagrams which they draw on the blackboard, and by formulae and text which they write on the blackboard. This scene is depicted schematically in fig. 1.

Each expert makes a contribution to the problem solving process. The experts can put items on the blackboard, they can erase items from the blackboard, or they can refine items which have been placed there by someone else. As items are placed on the blackboard, they can begin to form parts of an overall picture which might lead to a solution to the problem (even if they are unable to find one). These collections of parts of the solution are frequently termed "islands". As islands develop they are linked together, perhaps by arrows or some other form of indexing or cross-referencing. Eventually a solution might develop, probably amidst some kind of 'spaghetti' of arrows caused by the fact that the experts put ideas, hypotheses, etc., on the board in the most convenient place.



The fundamental blackboard model

Fig. 1

An important point to note about the metaphor is that each expert is totally independent of the others. Each expert does not rely upon the others for his knowledge, nor for his intervention in the problem solving process. The interventions of an individual expert depend upon what is on the board (at least, for the purposes of the metaphor). When there is something on the board to which at least one expert can respond, then he or she does so, modulo social conventions.

The interaction of the experts described in the last paragraph is an example of opportunism in problem solving. Each expert contributes something to the solution as, and when, the item is needed: the particular item depends upon the state of the solution, and the expert responds to the state of the solution by placing the item on the board. Opportunism can work by the detection of significant patterns in the current solution state and responding to it immediately. A final point to make about the experts' interactions is that they can be either serial or parallel: the experts can put items on the board either one at a time, or two or more experts can put them on the board at the same time.

The account of the experts' problem solving activity given above is, however, somewhat misleading. It is particularly misleading in its implication that the experts merely respond to random events on the blackboard. If this were truly the case, then achieving a coherent solution would be a very hit-and-miss phenomenon. Instead, the experts tend to concentrate on particular aspects of the problem and particular parts of the developing solution. In other words, they focus their attention in an effective manner. In any particular case, the experts will focus on the more promising, or what they perceive to be the more important, features of the problem on the current state of the solution.

Having concentrated on a set of particular features, islands or strategies, it is also possible to see experts relate islands or features which suddenly appear important. This is another example of opportunism, and indicates that problem solving strategies are not adhered to consistently throughout the solution process. This phenomenon has been noted by the Hayes-Roths ([Hayes-Roth, 1979], [Hayes-Roth, 1984]).

The significant points of the blackboard model have now been presented. The relationship between the model and its various implementations will be discussed in the next section.

Having now discussed the model in metaphorical terms, it is now time to look at various implementations and the architectural concepts they employ. For the purposes of the implementated systems one must consider the experts as being deaf and dumb: they are able only to communicate by placing information on the blackboard and not by talking to each other.

3. BLACKBOARD IMPLEMENTATIONS

In this section, the abstract model of the blackboard is transformed into a set of architectural principles shared by the majority of implemented systems. Once the mapping between the model and architecture has been established, a number of example systems are described.

3.1. Mapping between Model and Architectures

The model described in the last section contains four main elements:

- a database used by all the experts (the blackboard), and
- the experts themselves;
- items on the blackboard (diagrams, formulae, words, etc.), and
- links between items on the blackboard (arrows, numbers, etc.).

The blackboard represents a database in which records of the steps contributing to the solution are recorded. The database contains items which are placed there by the experts. The records represent states in the solution space which have been explored by the experts. When the problem has been solved, the items on the blackboard represent all the attempts by the experts to converge on that solution.

Although the physical blackboards left behind by human experts seldom show signs of hierarchy, a large proportion of complex problem domains can often be refined on the basis of hierarchies (for example, planning ([Sacerdoti, 1974], [Sacerdoti, 1977]).) The conventional approach to the construction of blackboard systems is to construct a hierarchical organisation for the database based on abstraction, as will be seen in subsequent sub-sections of this paper. (The reader should note that hierarchical decomposition of the blackboard database is not a necessity.) Within each level in the blackboard hierarchy, items are represented as data structures which are variously termed hypotheses or entries. Each entry

(the term preferred in this dissertation) is a complex structure which contains links to other items on the blackboard at the same, or at different, levels of abstraction within the hierarchy.

Entries are placed on the blackboard by Knowledge Sources (KSs) which are the analogues of the experts in the model.

Knowledge Sources are intended to play the role in the architecture which is played by the experts in the model. Knowledge Sources respond to the state of the blackboard by placing new entries on it. Knowledge Sources are activated by some kind of pattern-matching operation (which can be coded predicates which are specific to some application, or can be represented using a pattern-match syntax). If the entries on the blackboard are such that some of the KSs can respond, then they are eligible to make some response. Typical responses for KSs are:

- putting a new entry on the blackboard;
- deleting an entry;
- modifying an entry (by, say, adding a link or set of links connecting it to some other entries); and
- causing an external event (such as requesting some information or outputting a result to some other object.)

KSs are normally represented as a pair consisting of triggering-pattern (that which responds to blackboard entries or changes in configuration on the blackboard - the latter is determined by looking for specific links between entries, looking for entries with specific properties or data elements or can be performed on a basis analogous to interrupt processing) and an action-part. The action-part is responsible for performing operations of the kind listed above.

KSs are intended to represent experts. This entails that they should be comprehensive in their function and probably quite large. In HEARSAY-II (see section 3.2 below), KSs were very large indeed, but there appears to be a tendency to reduce the size of KSs (e.g. OPM - section 3.3) so that their triggering patterns can be made fuller and their effects on the blackboard reduced in scope.

Although large KSs lead to opaque chunks of code in a blackboard system, they reduce the control overheads just because they are none too specific in their triggering conotations. Small KSs, on the

other hand, increase control load (by weight of numbers) and lead to a kind of fragmentation: many KSs may be related to a small set of conditions, but are logically distinct entities which ought to be considered at the same time, but since they are not collected into logical modules, the space of all KSs must be searched for those which can be considered given prevailing circumstances. This point appears to be crucial when considering sophisticated control regimes.

Large KSs also entail that there are possibly many scheduling factors which ought to be taken into account. Smaller KSs tend to lead to fewer parameters just because small KSs are more specific and more easily defined. The reason for this is that large KSs must represent entire classes of solution (as in HEARSAY-II (sect. 3.2) whose KSs produced *all* plausible hypotheses on the basis of one input): thus, each of the candidate solutions which *might* be produced by a KS has to be taken into account when scheduling. Also, if a KS has a large grain size, it can, possibly, be activated on a number of regions of the blackboard (one very often finds that what was designed to be a single level on the blackboard, in reality, represents a collection of abstraction levels, each one of which becomes a region for the purposes of side-effects and matching), and, again, these must be taken into account when constructing the scheduler. From these considerations, it can be seen that the number of scheduling parameters tends to increase with the grain size of the KSs in the system.

In the following sub-sections, a number of blackboard systems will be described. Rather than convert the terminology into a neutral form, the terms used in the original papers are used. It is necessary, therefore, to introduce some equivalences between terms. The Hearsay-II term 'hypothesis' is referred to as an 'entry' in OPM and Ariadne-1; the conventional term 'KS' is changed to 'specialist' in the Hayes-Roths' 1979 OPM planning system. The Hayes-Roths also use the term 'partition' where others use the term 'panel'.

3.2. HEARSAY-II

The HEARSAY-II continuous speech understanding system ([Erman, 1975]* and [Hayes-Roth, 1977]) was the second and, perhaps the best known implementation of the blackboard model. The task

* This description is mostly taken from Erman.

domain of HEARSAY-II was the understanding of continuous speech in approximately real time.** The blackboard model was chosen for HEARSAY-II because of the large problem space and the large amount of knowledge required for the solution of the problem. Furthermore, the knowledge needed for the domain is diverse, including signal analysis, phonology, syntax and semantics. Other factors included the problem that the knowledge provided in the system may contain errors or may be incomplete (either theoretically, or because the implementation is incomplete or erroneous). This problem requires that results must be classified as soon as possible and errors reduced: this entails that items of knowledge should co-operate in the solution. Furthermore, the limitations of human knowledge for the domain entailed that a high degree of modularity was required, both in the representation of knowledge and in the strategies used to control the application of that knowledge.

The global database of HEARSAY-II is partitioned into distinct information levels. Each level holds a different representation of the problem space. The levels in the blackboard are organized in a loose hierarchy: the hierarchy is based upon abstraction.*** The lowest level is concerned with digitized signal processing (this is termed the parametric level); the highest level is the conceptual level which deals with the concepts used in the utterance being analysed. Within each level, there are hypotheses. Hypotheses correspond to the 'entries' of the previous sub-section. Each hypothesis is composed of a set of primitive elements associated with a particular level (for example, the lexical level contains words from the lexicon as its primitive entities). Hypotheses exist only at particular levels and are labelled as being some particular element of the set of primitive entities for that level. The level structure of the HEARSAY-II blackboard is shown in fig. 2 (from [Erman, 1975]).

The decomposition of levels in the blackboard represents a natural decomposition of the problem space: it also parallels the decomposition of Knowledge Sources (see below). The loosely hierarchical structure of the blackboard leads to the description of a level as being an abstraction of levels lower than it in the hierarchy. The abstraction mechanism can be exemplified as follows: a lexical item (word) is composed of syllables; syllables are composed of phonemes, each of which is composed of acoustic

** The domain was the language used extract computer science abstracts from a database.
*** Most implementations have retained the abstraction hierarchy for various reasons -- e.g., see the comments on the SU programs below (sect. 3.5).

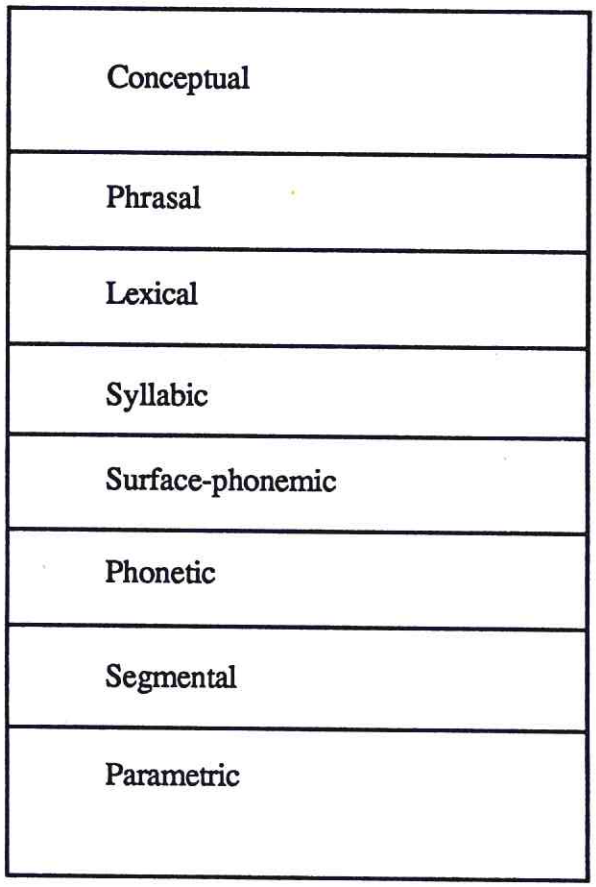
segments.

Hypotheses on the blackboard are connected by numerically weighted links to form a network: the numerical weighting represents the strength (i.e., the plausibility) of that link. The majority of the relationships which exist are between hypotheses at the same level of abstractions. Speech understanding depends upon linear sequences of items (for example the sequence of lexemes is crucial in non-inflected languages such as English): therefore, many links represent the ordering of items with respect to time within a level (for example, at the parametric level, hypotheses are ordered by the arrival time of the signal components). Other kinds of link exist: these usually relate an hypothesis at one level of abstraction with hypotheses at other, typically adjacent levels. The hypotheses and the levels on which they occur, form a context within which knowledge sources can activate. A fragment of the HEARSAY-II blackboard is shown in fig. 3 (from [Erman, 1975]).

Hypotheses, as was noted above, are structured objects. The goal of HEARSAY-II is to form plausible hypotheses from the input signal. Hypotheses contain a field containing the rating of the hypothesis. The rating of an hypothesis is a universal measure of the plausibility of the hypothesis (similar to confidence factors in MYCIN ([Shortliffe, 1976])) and is derived from a number of sources such as a priori information about the hypothesis, information derived from hypotheses at a lower level (synthesis) or information derived from hypotheses at a higher level (analysis). There are also fields for recording the amount of computational resources expended on an hypothesis.

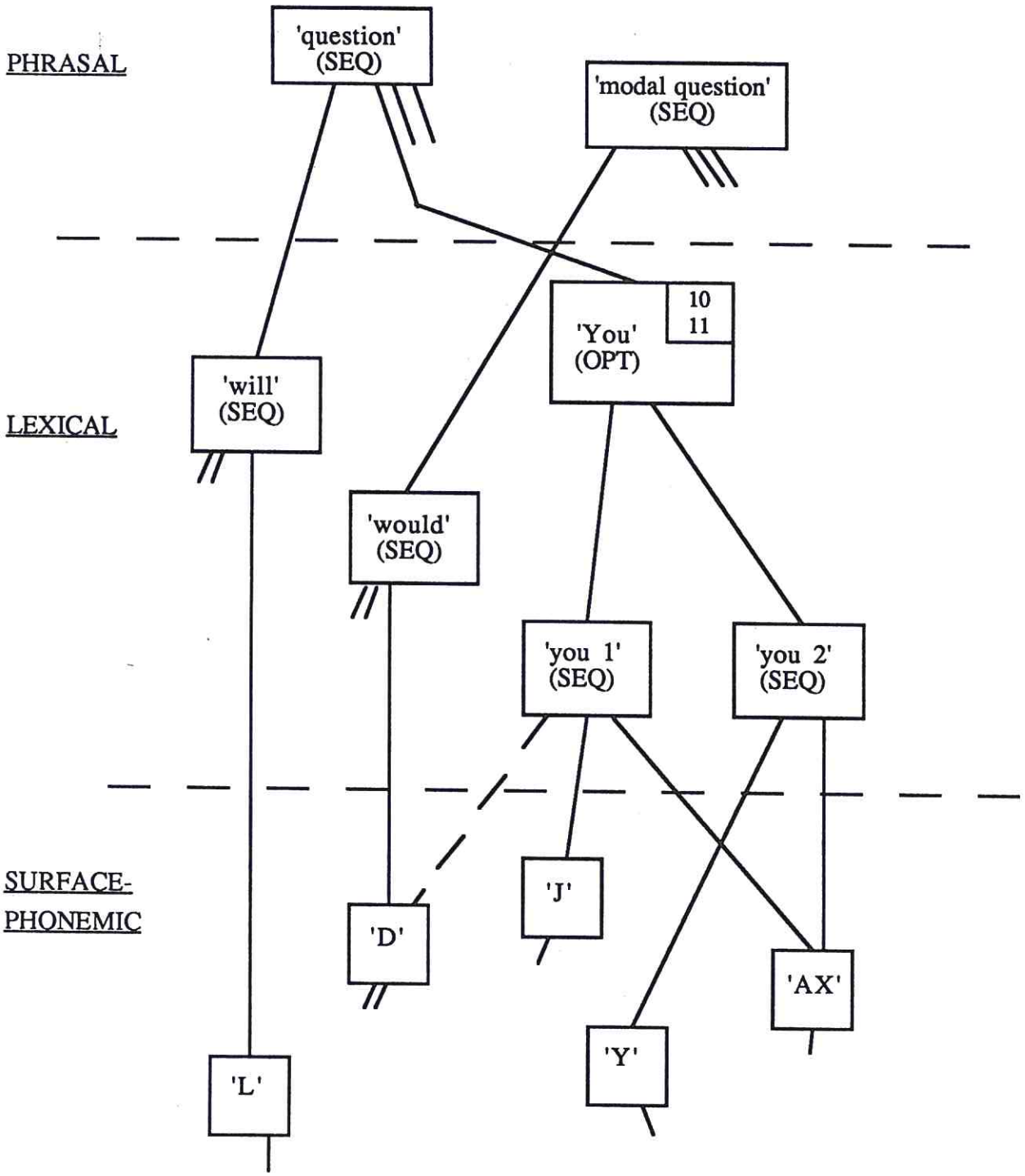
The HEARSAY-II system has been described ([Erman, 1975]) as a "multi-level organization for problem solving using many, diverse, co-operating sources of knowledge". In the system, hypotheses represent short-term memory structures, whereas long-term memory structure is represented as Knowledge Sources (KSs).

Knowledge Sources in HEARSAY-II parallel the decomposition of the blackboard: they partition the problem space into logically distinct areas of expertise in a natural fashion. KSs are usually associated with one or two levels on the blackboard. Each KS can access any information on the blackboard and can update the blackboard by adding, deleting or updating hypotheses and by updating (etc.) links. KSs are viewed as being totally independent items of knowledge which communicate with each



Abstraction levels in HEARSAY-II

Fig. 2



A fragment of the HEARSAY-II blackboard

Fig. 3

other only through the hypotheses on the blackboard. This constraint entails that each KSs is not aware of the existence of other KSs. The addition or removal of KSs can not affect the other KSs in the system. The Knowledge Sources in the HEARSAY-II system in 1975 are shown in fig. 4. The figure shows the interactions between Knowledge Sources and the levels on which they operate.

The high degree of modularity inherent in this approach was one of the design aims of the system. As a strategy for building systems such as HEARSAY-II, the blackboard model requires experimentation with both the knowledge content of the system and the strategies used to control the knowledge in the system. (The control aspects of HEARSAY-II will be described below).

Knowledge Sources in HEARSAY-II have three components:

- the conditions under which they can be activated. This is expressed in terms of the configuration of hypotheses on the blackboard which causes the KS condition to become matched, or the single hypothesis on the blackboard which uniquely matches a KS condition.
- the kinds of changes the KS makes to the blackboard.
- a procedural statement of the algorithm which makes the changes (this is a piece of SAIL code).

KSs possess the computational ability to solve subproblems given appropriate conditions on the blackboard.

KSs are activated and instantiated whenever the blackboard exhibits characteristics which satisfy a precondition of the KS. A KS precondition is a description of a partial state of the blackboard which defines when and where a KS can make a contribution to the problem solving process (the location is specified in the stimulus-response frame associated with each KS). In HEARSAY-II, contributions are made relative to a context: a context is a subset of the previously generated hypotheses already on the blackboard. Modifications caused by a KS are used to trigger further KS activations by creating new conditions to which other KSs can respond. KSs run in data-directed mode: the data which causes activation are the hypotheses on the blackboard. This entails that KSs can also exhibit a high degree of asynchronous activity and potential concurrency.

In HEARSAY-II, KSs are associated frequently with only a very few abstraction levels, some-

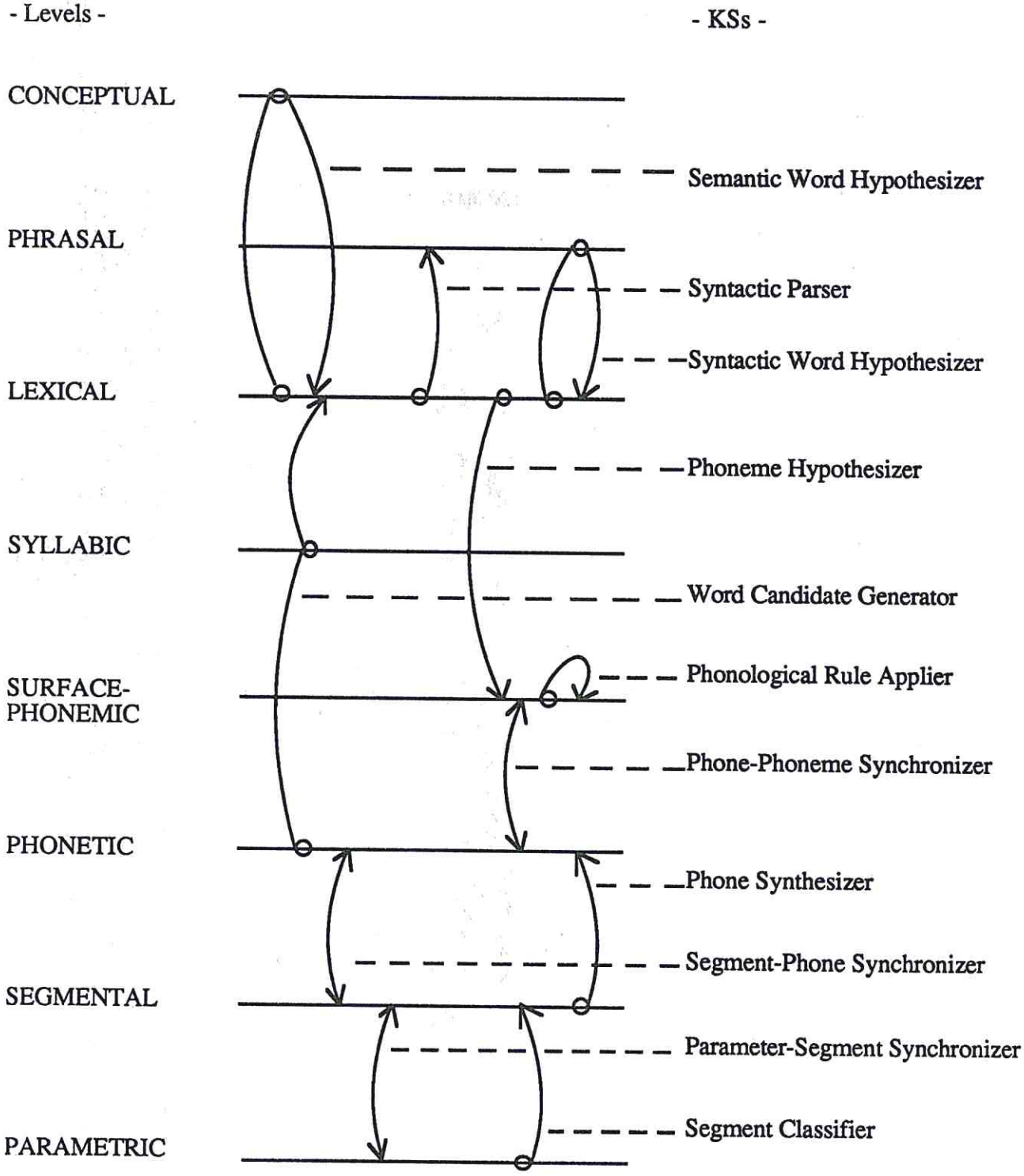
times only one. Given the modularity of the KSs, conceptually KSs need only be aware of the levels with which they interact directly: all other levels are, in effect, invisible to them. Thus, the preconditions of a KS are restricted in the range of objects which can activate them. The precondition evaluation in HEARSAY-II is made more efficient by placing functions in the routines which update the blackboard. These functions are activated when a KS modifies an attribute of a blackboard hypothesis which some other KS(s) has (have) asked to be monitored. This data is held in a structure called the change set which is specific to the monitored attribute and the KS responsible for the monitoring. KSs can specify either statically or dynamically those attributes they may wish to have monitored. Monitoring can be restricted to individual levels or hypotheses.

Change sets categorize events (blackboard modifications) and are useful in precondition evaluation because they limit the areas of the blackboard which need to be examined in detail.

HEARSAY-II is a goal-directed system and employs a two-stage scheduler to control KS activation. The goal of the system is to produce the most plausible network of hypotheses from the input data. At any stage, there will be incomplete networks of hypotheses which possibly share sub-networks. During processing, networks can be extended or contracted. At any one time, there can be many KSs which can be activated. The HEARSAY-II system uses a goal-directed scheduler to decide which KSs to activate.

The scheduler uses some of the hypothesis attributes in making decisions, for example there are attention attributes which state how much processor time has been used by the system attending to the hypothesis. KSs can also suggest how desirable it is to devote effort on an hypothesis. Processing state attributes are also used by the scheduler.

The scheduler can also use focussing factors which highlight activity in areas of the blackboard in which there are no links between adjacent levels. In such cases, the scheduler should give higher priority to a KS which attempts to make such connections (this is indicated in the KS's external specifications) than to a KS which makes minor alterations to hypotheses. Alternatively, a KS may be selected because it is the only one which can be run. A further factor which determines activation priority is the validity of the hypotheses in the area of the blackboard which satisfied the preconditions of KSs. This



HEARSAY-II Knowledge Sources (1975)

Fig. 4

last data can be used to implement a best-first strategy.

The implementation of goal-directed scheduling is decoupled from the actions of individual KSs. The decoupling of KSs and scheduling leads to a quick refocussing of KS attention. Rapid refocussing is important to HEARSAY-II because of the errorful nature of KS activity which leads to incomplete and possibly contradictory hypotheses.

Focussing factors are goals (attention attributes) attached to hypotheses. Goals indicate the kind of change desired in a hypothesis attribute and the desirability of the change. HEARSAY-II permits goals to be attached to regions of the blackboard and not just hypotheses. Hayes-Roth and Lesser ([Hayes-Roth, 1977]) describe in detail a more detailed and general focussing mechanism for HEARSAY-II. I will return to the HEARSAY-II scheduler when discussing OPM (section 3.3, below).

3.3. OPM

The blackboard system described in ([Hayes-Roth, 1979]) represents a development of the HEARSAY-II implementation. The blackboard develops HEARSAY-II by introducing extra structural partitions into the model, by explicitly representing control information and by simplifying control.

The extra architectural divisions of the blackboard and the explicit representation of control make this system particularly interesting.

The task domain of the system is errand-planning. The knowledge embodied in the system is derived from protocols taken from human subjects who were given a set of tasks to perform (such as 'get medication for the dog') in the context of an imaginary town. The decomposition of the planning problem given by the Hayes-Roths is that it is composed of two primary components: the first is the assembly of planning operators; the second is the execution of the plan. They refer to this paradigm as planning and control. However, unlike Sacerdoti's NOAH system ([Sacerdoti, 1974]), the approach they advocate is not, basically, hierarchical, but opportunistic. (Opportunism was described in section 2 as a motivating factor for the blackboard model). The Hayes-Roths do make the point that opportunism is not incompatible with refinement planning (this is because they are both, basically, control strategies): this point will be explained in greater detail in the next section when OPM is described.

In the context of the 1979 system, the Hayes-Roths describe opportunism as follows:

"However, we assume that people's activity is largely opportunistic. That is, at each point in the process, the planner's current decisions and observations suggest various opportunities for plan development. The planner's subsequent decisions follow up on selected opportunities. Sometimes, these decision sequences follow an orderly path and produce a neat top-down expansion as described above. However, some decisions and observations might also suggest less orderly opportunities for plan development."

([Hayes-Roth, 1979], p.276).

The resulting system uses an opportunistic model of control. The knowledge in the system is derived from the experimentally obtained human protocols. The knowledge derived from the protocols is encoded as knowledge sources which are called specialists ([Hayes-Roth, 1979], pp 285,286).

The specialists make tentative decisions for incorporation in a tentative plan. Specialists are differentiated along a number of dimensions, one of which is abstraction - some specialists suggest high-level, abstract additions to the plan; others make detailed sequences of specific actions. Specialists are described in the simulation system as generalised condition-action pairs (similar to the trigger-pattern/action-part description KS in section 2.1). Specialists respond either to planning decisions already made, but they can also respond to configurations of decisions, just as HEARSAY-II Knowledge Sources can be activated by the presence or absence of links between hypotheses. The action component of specialists can perform arbitrary amounts of computation, but must result in the creation or modification of decisions on the blackboard.

As with HEARSAY-II, specialists are associated with abstraction levels on the blackboard. The planning system, however, introduces extra structural divisions, called planes, on the blackboard. The planning system has a blackboard with five planes: plan, plan-abstractions, knowledge base, executive and meta-plan. Each plane is concerned with a specific kind of function within the system. The structural divisions are motivated by the need to separate the functions and to provide facilities to perform each of the functions independent of each other. Each plane is divided into abstraction levels: in this respect, each plane operates as a complete HEARSAY-II blackboard, but communication between planes is

always possible (by means of specialists).

The plan plane contains items which represent those actions the planner intends to take in the world. In the context of the imaginary town used in the experiment, the planner might want to travel round in a circle or take the shortest route between all the places on the list of errands.

Decisions on the plan-abstraction plane characterise the properties of plan decisions (on the plan plane) which are made by the planner. These decisions indicate the actions the planner wants to take, but without specifying the actions in any kind of detail. Decisions in this plane represent descriptions of sequences of actions. Descriptions of action sequences do not identify particular sequences of actions which need to be taken. Alternatively, the planner may decide to organise the plan around a spatial cluster or errands: this is, again, abstract and does not specify particular actions.

The knowledge-base plane contains observations and computations about relationships in the world of the imaginary town which might have some relevance for the planning process. These items are useful in suggesting plan-abstractions and in initiating plan abstractions in the resulting plan.

The combination of plan, plan-abstraction and knowledge-base provide a context within which to determine features of the plan. The executive plane determines features of the planning process. The executive plane contains decisions about the allocation of resources during the planning process. The decisions made by the executive determine which aspects of the plan are to be developed and which specialists are to be employed at any given point in the planning process. For example, the executive might decide to determine which errands are to be included in the plan before producing any details of the plan, or it might decide to focus on finding routes among previously sequenced errands.

The meta-plan plane contains decisions about how the planning problem should be approached. The decisions made by the meta-plan plane reflect the planner's understanding of the problem, the methods to apply to the planning problem and the criteria to be applied in the evaluation of the various solutions derived by other planes.

As was stated above, the model adopted by the Hayes-Roths divides each plane into levels of abstraction, in a manner very much like the division of the HEARSAY-II blackboard. It is therefore possible to consider the abstraction levels which sub-divide the problem space represented by each plane.

The levels in each plane form a potential hierarchy.

In the plan plane, the four levels represent a refinement hierarchy, each level of which represents a more abstract version of the developing plan than the level(s) which is (are) below it. The most abstract level expresses the outcomes that the planner expects from the plan; the most concrete level is the operation level which specifies the sequence of actions which are to be realised in the world. Between these levels are the designs level and the procedure level, in decreasing order of abstraction. The designs level characterises the general approach to be taken by the planner to achieve the desired outcomes. Specific sequences of actions (which are refined by the operations level) are defined by the procedures plane: in the errand planning domain, procedures specify sequences of errands.

The plan-abstraction plane also has four levels. Each level contains characteristics of the type of decisions to be incorporated in the plan. There is a correspondence between the plan-abstraction plane and the plan plane: decisions made on each level of the plan-abstraction plane map directly onto the more concrete decisions made on the corresponding level of the plan plane. Within the plan-abstraction plane, intentions are indicated at the highest level: these map onto the outcomes of the plan plane. Intentions establish all of the initial errands to be performed. Below intentions come schemes which suggest designs featuring spatial clusters of errands in the plan. Next, the planner might develop strategies to schedule errands; below this, come tactics which look for short-cuts between errands. Tactics determine desirable operation level decisions in the plan plane.

The knowledge base panel has four levels. The knowledge base plane contains problem-specific information. The four levels are errand, layout, neighbour, and route. Each level contains observations and computations required to perform the corresponding operations in the plan and plan-abstraction planes. Information contained in the knowledge base plane is derived from the topography of the town in which the errands are to be performed.

In contrast to the three planes already discussed, the executive plane has only three levels. The inclusion of the executive plane is the second of the interesting points about this system. The executive plane performs a hierarchical control task: each level refines the level above it. At the highest level, the plane makes decisions about the priorities of various requests for resources. Priority decisions indicate

where resources should be allocated: this acts as a focussing mechanism which directs attention to some areas of the plan plane before attention is directed to others. The next level is the focus level. The focus level makes decisions which indicate the decisions to be made at any specific point in the planning process. Finally, schedule decisions are made to resolve conflicts between scheduling decisions and are made in a manner similar to those used in the later work on HEARSAY-II ([Hayes-Roth, 1977]). The impact of a separate division of the blackboard will be discussed in greater detail in the next section.

The final plane is the meta-plan plane, which has four levels. Unlike the other planes, the meta-plan plane does not divide into a neat hierarchy: instead, it emphasises different aspects of the human subject's approach to the planning problem. The levels on this plane relate in systematic ways to the other planes of the blackboard. Since this plane is highly-subject specific, no further description will be given.

The layout of the Hayes-Roth's blackboard is shown in fig. 5. The figure also shows some illustrative actions of specialists (KSs).

OPM is of considerable interest because of its division of the global database into planes, each of which is concerned with a specific kind of cognitive process. In their paper (Hayes-Roth, op.cit.), the Hayes-Roths state that the division into planes is somewhat arbitrary, but is justifiable on the basis of subjects explanations of the kinds of knowledge they apply in the planning process. The importance of the system for this paper is that it emphasises not only the flexibility of the blackboard model, but also the fact that fine distinctions in the knowledge used to solve problems can be reflected in the structure of the problem solving system. Thus the errand planning system distinguishes plans from plan-abstractions and meta-plans (which have also been investigated by Wilensky ([Wilensky, 1983]) and Schank ([Schank, 1982])): the three planning-specific planes are kept separate from knowledge about the environment in which the errands are to be performed. Finally, the four planes just mentioned are separated from the executive plane which controls the system by means of our explicit representation of control information.

The second point about the system is the executive panel. The executive panel controls the operation of the system. It controls the planning actions of the system. Control decisions are

represented explicitly in the executive plane. Explicit representation of control implies that the system can reason about control. Control is thus amenable to intelligent processing, and becomes another domain in which problems can be solved. In the next section, when considering the Blackboard Control Model and Hearsay-III the benefits of explicit control will be examined in greater detail.

3.4. Hearsay-III and the Blackboard Control Model

In this section, Hearsay-III ([Balzer, 1980], [Erman, 1981]) and the Blackboard Control Model ([Hayes-Roth, 1983a], [Hayes-Roth, 1984], [Hayes-Roth, 1985]) are described. The two systems are grouped together for the following reasons:

1. Both systems basically follow the HEARSAY-II architecture;
2. Both systems contain structural divisions for the explicit control of domain knowledge (as is the case in the Hayes-Roth's system described in the last section).
3. Both systems represent considerable developments of the original HEARSAY-II design.

This section is divided into two parts. The first part describes Hearsay-III. Then the Control Model will be described. The discussion of the Control Model centres on a description of the control structure (rather than on particular implementations) and alternative control regimes for HEARSAY-II and the HASP sonar interpretation system ([Feigenbaum, 1982]).

3.4.1. Hearsay-III

Hearsay-III ([Balzer, 1980], [Erman, 1981]) is designed to be a flexible tool for the construction of knowledge-based or expert systems (it is a shell, similar in spirit to EMYCIN ([van Melle, 1979]), and ([Kunz, 1983]).)

Hearsay-III is implemented on top of a relational database system, called AP3 ([Goldman, 1978]), from which it inherits strong typing and control facilities. The use of AP3 as a base for Hearsay-III is one of the distinguishing features of Hearsay-III's implementation. AP3 has similarities with the PLANNER-like languages ([Hewitt, 1972]) and Prolog ([Clocksin, 1982], [Kowalski, 1979], [Warren, 1977] [Colmerauer, 1975]), but it also provides strong typing for assertions, retrievals and argument

transmission in function calls (similar features have been proposed for Prolog by Mycroft ([Mycroft, 1984])). All of the publicly accessible data structures in Hearsay-III are represented as structures in the AP3 database. AP3 also provides demons* which are used by Hearsay-III Knowledge Source triggering patterns. Because triggering patterns in knowledge sources are represented as demons, KSs can respond easily to changes in the blackboard.

AP3 also provides a context mechanism similar to that found in CONNIVER ([McDermott, 1974]). Hearsay-III makes the context mechanism available to user applications, and also makes it an integrated part of the reasoning mechanism. Contexts allow reasoning along independent paths which can be generated by competing knowledge sources and by choices among competing solutions.

Finally, AP3 provides facilities for inference rules and constraints, both of which are available to Hearsay-III users for the representation of global domain-dependent relationships. Inference rules and constraints are also used in the implementation of Hearsay-III ([Erman, 1981]).

The facilities afforded by the AP3 relational database system were, at the time Hearsay-III was built, quite unique in a shell.

In Hearsay-III, as in all of the systems described in this paper, the central communications medium is the blackboard. However, in Hearsay-III, the structures appearing on the blackboard are quite different.

The Hearsay-III blackboard contains a graph representing partial solutions and pending actions. Partial solutions and pending actions are represented as nodes in the graph. Nodes on the blackboard have internal structure, and are referred to as units: the arcs between nodes are called roles. The blackboard in Hearsay-III is initially divided into two parts, called the domain blackboard (which represents partial results and pending actions specific to the problem domain), and the scheduling blackboard. The scheduling blackboard is used to control domain and control knowledge. Control knowledge is represented in the form of knowledge sources (Hearsay-III Knowledge Sources are slightly different from conventional KSs as will be described below).

---

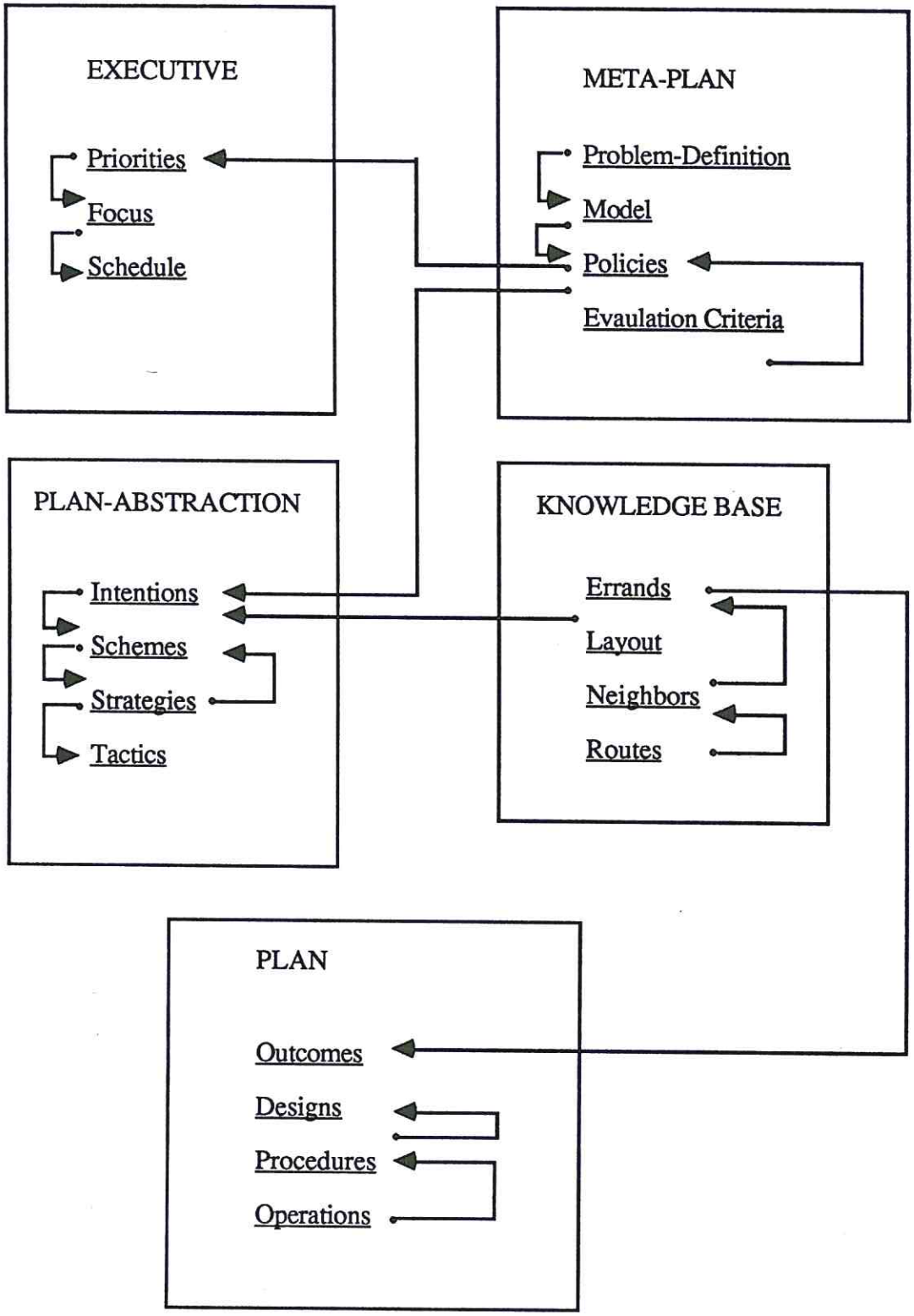
* A demon is a procedure which is activated in an asynchronous fashion when some condition obtains in the 'world' which the demon is monitoring.

The two blackboards mentioned above are the minimum divisions of Hearsay-III. Application builders can further subdivide the blackboard as necessary. This subdivision can be achieved by use of the class structure of the system.

The units which represent nodes in the blackboard graph are implemented as typed objects in AP3. The types are referred to as unit-classes. The division of the blackboard into logically distinct spaces is achieved by decomposing the most general class into several, distinct, subclasses. The domain blackboard is composed of the class Domain-Unit and its subclasses; the scheduling blackboard is composed, in the same way, of the class Scheduling Unit. Access is restricted to particular classes by means of the AP3 typing mechanism. User-defined classes can be used to introduce finer divisions of the blackboard.

In addition to their types, units have internal structure. An interesting and important feature of unit structure is that they can represent unresolved decisions. Units of this kind are called choice sets. Choice sets are associated with a set of alternatives which can be static (i.e. declaratively represented), or dynamic (i.e., represented by generators of alternatives). Choice sets can be thought of as the partial elaboration of a decision: the alternatives represent further, potential, elaborations of the decision. Alternatives can, themselves, be choice sets. The choice set permits competing solutions to spread from a single point on the blackboard.

Since choice sets represent ambiguities in the solution space generated by the knowledge in an application system, mechanisms must be provided to resolve the ambiguities. Hearsay-III provides two mechanisms to do this. The first mechanism is called deduce-mode Choose: this can be used when the application has conclusive evidence that one alternative is the correct solution for the problem represented by the choice set. The deduce-mode Choose is backed up by the fact that the application will not retract the choice it has made when further evidence comes to light. When a deduce-mode choose is made, the choice set is replaced by the selected alternative in the context in which the choice is made (all decisions are made within contexts). When the alternative is selected, all evidence of the choice set is removed from the context in which it occurred (so that no backtracking or retraction can be performed - deduce mode removes non-determinism).



The planning blackboard (Hayes-Roth & Hayes-Roth, 1979)
(levels are underlined; KS actions are indicated by arrows)

Fig. 5

The second choice mechanism is called assume-mode Choose. An assume-mode Choose replaces the choice set by a unit which represents the merge of the choices, as is the case in deduce-mode. Assume-mode, however, makes changes in the contexts on the blackboard: a new context is created as a child of the context in which the choice was made (and thereby inherits all the units in the parent context, its parent, and so on). The structure in the new context is identical to that produced by deduce-mode. The choice-set still exists in the parent context, but it has one alternative removed: the alternative which is removed is the one which was selected to become part of the new child context. If subsequent inference indicates that the choice was not the best, it can be retracted, and focus can return to the parent context, and another alternative selected. Assume-mode choice permits both backtracking and the removal of assumptions: it permits backtracking in a variety of forms, such as dependency-directed backtracking ([Sussman, 1977]). Assume-mode choose could be used to build a system using Truth Maintenance ([Doyle, 1979]); also, it introduces a measure of non-determinism into inference.

Units have a further attribute associated with them: this is called an acceptance. Acceptance is the process of assimilating a unit into a larger structure and verifying that it is appropriate in that structure. Hearsay-III permits the application-builder to associate a LISP predicate with each unit class. The predicates perform the functions of validation, canonisation (determining whether the unit is in canonical form for the larger structure), uniqueness-determinination, conflict-determination and integration. Each time a unit is created or is marked as unaccepted by a Knowledge Source, the acceptance routines for the unit's class are executed: if they all return true, the unit is marked accepted, otherwise, it is marked unaccepted, and the unit's context is marked as *poisoned* (see below). Hearsay-III prevents KSs from triggering on unaccepted units, since they are not certain to contribute to the solution of the current problem.

As was stated above, units have arcs which connect them with other units in the graph on the blackboard. The arcs are called roles or component roles. Roles in Hearsay-III are typed relations, and this considerably simplifies the searching of the structure when retrievals are performed. Roles belong to classes, called role sets. Role sets can be used for two purposes. Firstly, role sets define logically distinct component hierarchies in which units can be represented: the hierarchy is defined as the transitive clo-

sure of all the roles in the role set. The hierarchies defined over roles act to generate abstractions of the blackboard structure.

The second use of role sets is the description of mutually exclusive units in aggregated structures on the blackboard. Mutual exclusion works by prohibiting structures in which there are two units which are both components of a third unit, both of which have a unique fourth unit in common. This prohibition entails that partial solutions should not be interpreted in different ways in a global solution.

A number of references have been made to the context mechanism in Hearsay-III: it is now time to consider it in more detail.

Choice sets provide a means for representing unmade decisions about alternative interpretations. However, on their own, choice sets are inadequate for they are unable to prohibit interference between investigations into alternatives. The extra structure is provided by the context mechanism. Contexts were invoked when describing deduce- and assume-mode choices, but the relationship between context and the control of inference was not made explicit.

Contexts in Hearsay-III are organised into a hierarchy in a manner similar to units. They are used to present reasoning in logically distinct subspaces of the entire solution space without interference from other, concurrent, reasoning attempts. Reasoning in Hearsay-III, as in all implementations of the blackboard model is performed by Knowledge Sources (KSs), and there is a relationship between KS triggering and contexts: KSs trigger in the most general context in which its pattern is matched. When the action-part of the KS is executed, it performs actions in the context in which its trigger evaluated to true (unless the KS explicitly changes the context in which it is to act). When actions have been made in a context, the results of those actions are inherited by all sub-contexts of the context in which the action was performed (i.e., all contexts which are closer to the leaves of the tree of contexts which have the context in which the actions were performed as its root).

Contexts can be marked as being unsuitable for executing KSs. This condition is termed poisoned. Poisoned contexts result from the violation of a Hearsay-III constraint, for example trying to aggregate conflicting units. A KS can poison contexts, if, for example, it discovers a violated domain constraint. If a KS has been activated or triggered in a poisoned context, it is put into a wait state until the context is

unpoisoned. Hearsay-III permits special KSs, called poison handlers, to run in poisoned contexts. Poison handlers are intended to diagnose and correct the problems which caused the poisoning in the first place.

Knowledge Sources (KSs) in Hearsay-III are another component of the system which is interestingly different to the other systems described in this paper. As in other blackboard systems, Hearsay-III KSs represent long term memory or the experts in the blackboard model. KSs react to changes on the blackboard and can perform all the functions normally associated with KSs. However, the extra structure of the blackboard and the implementation of Hearsay-III in AP3 entail some significant differences in Hearsay-III KSs.

The structure of Hearsay-III Knowledge Sources differs from the structure of HEARSAY-II KSs. In Hearsay-III, KSs have a triggering-pattern, a piece of immediate code and a body. Whenever the triggering-pattern matches an object on the blackboard, Hearsay-III creates a unit, called an activation record, and runs the immediate code. At some later time, the activation record might be chosen by the scheduler and executed. KS execution means that the body is run: the body is an arbitrary piece of LISP.

The KS activation process, in more detail, is thus:

- The triggering pattern is a pattern in AP3. The triggering pattern is a predicate, whose primitives can be AP3 templates and arbitrary LISP predicates composed with the logical operators 'and', 'or' and 'not' and is modified such that any AP3 templates in a pattern are satisfied, the entire KS triggering-pattern is evaluated. If the entire pattern matches, then an activation record is created. Activation records contain the name of the triggered KS, the AP3 context in which the KS matched (in AP3, the least general context) and the variables instantiated by the match.
- When the activation record is created, the immediate code is run. Immediate code is composed of arbitrary LISP code: it can associate information with the activation record which may be of some value when selecting an activation record for execution. The immediate code returns, as its value, the name of some unit class on the scheduling blackboard. The activation record is placed on the scheduling blackboard as a unit of the class named by the immediate code. Immediate code is run in the triggering context of the KS and can access the instantiated pattern variables.

- Some time later, the scheduler calls on the Hearsay-III Execute action. The Execute action is applied to the activation record. The usual result of this is to run the body of the KS corresponding to the activation record. The body is run in the triggering context and has access to the instantiated pattern variables. If the triggering context is poisoned when the body is run, and the KS is not a poison handler, the body cannot be run. Instead, the activation record is marked as waiting for the context to become unpoisoned: if the context is ever unpoisoned, the activation record is marked ready (the status of activation records whose contexts are not poisoned).

In Hearsay-III, KS execution is indivisible. KSs run to completion and are not interrupted to allow other KSs to be run. This constraint simplifies the construction of KS bodies and the interactions permitted between KSs and units on the blackboard.

Hearsay-III is intended to be used in domains in which activation record selection (scheduling) is complex. This is why there is a scheduling blackboard in the system. The scheduling blackboard provides a medium for the explicit representation of control; it also allows application builders to experiment with control schemes. Scheduling decisions are represented as units on the scheduling blackboard (these units are activation records, amongst other things), and scheduling knowledge is represented as scheduling KSs.

The second phase of KS operation creates activation records. Activation records contain KS-specific information. Scheduling KSs can make additional changes to the scheduling blackboard: this contributes to the selection of activation records. Scheduling KSs are able to respond to changes on both domain and scheduling blackboards. The actions which scheduling KSs can take include associating information with activation records (for example, assigning and modifying priorities) and creating new units to represent meta-information about the domain blackboard (this might be setting priorities in the currently highest-rated unit in the domain blackboard).

Just as the domain blackboard is the database for domain problem solving, the scheduling blackboard is the database for control decisions.

In order to get the scheduling blackboard and its associated KSs to work, the application-builder must provide a base scheduler. The base scheduler is a LISP procedure which calls the Hearsay-III

Execute primitive to run selected KSs. The base scheduler is called after system startup. The base scheduler is intended to be as simple as possible since most of the knowledge about scheduling in the system ought to be expressed as KSs. One version of the base scheduler just removes the first element from a queue: should the queue become empty, the base scheduler just terminates - this denotes the end of the run.

Hearsay-III does provide a default base scheduler. It is constructed from two procedures: the outer base scheduler and the inner base scheduler. The outer base scheduler repeatedly calls the inner base scheduler. The inner base scheduler is expected to return a list of activation records. If the returned list is empty, the scheduler terminates. When the outer base scheduler gets a non-empty list, it runs each KS denoted by the activation records in turn. If the KS which is selected is a scheduling KS, its execution returns a list of non-scheduling KSs. The outer base scheduler executes each of these KSs in turn. When the inner base-scheduler is invoked, it selects one ready scheduling KS in a non-deterministic fashion.

The builders of Hearsay-III expect that the default inner base scheduler will be replaced by an application-specific scheduler. The outer base scheduler is expected to be a reasonable skeleton for many applications.

3.4.2. The Blackboard Control Model

The work described in ([Hayes-Roth, 1984], [Hayes-Roth, 1985]) grew out of work with the errand planning system ([Hayes-Roth, 1979]). The Control Model has, by now, been used in a variety of systems (e.g., PROTEAN ([Hayes-Roth, 1984b]), NBB ([Craig, 1987]), BB-1 ([Hayes-Roth, 1985a]) and BB* ([Hayes-Roth, 1986])). The points which Hayes-Roth makes in ([Hayes-Roth, 1984]) are that control knowledge can be represented explicitly and that explicit control can be applied to itself - it can be reflexive. The descriptions of the Control Model in ([Hayes-Roth, 1984]; [Hayes-Roth, 1985]) use the OPM system ([Hayes-Roth, 1979]) as a basis: the explanation of the functioning of the Control Model is given in terms of errand planning: I use the same context in this section.

The blackboard in ([Hayes-Roth 1984]) is divided into two non-overlapping sub-regions called panels. Panels correspond to planes in ([Hayes-Roth, 1979]). There is a panel for domain-specific

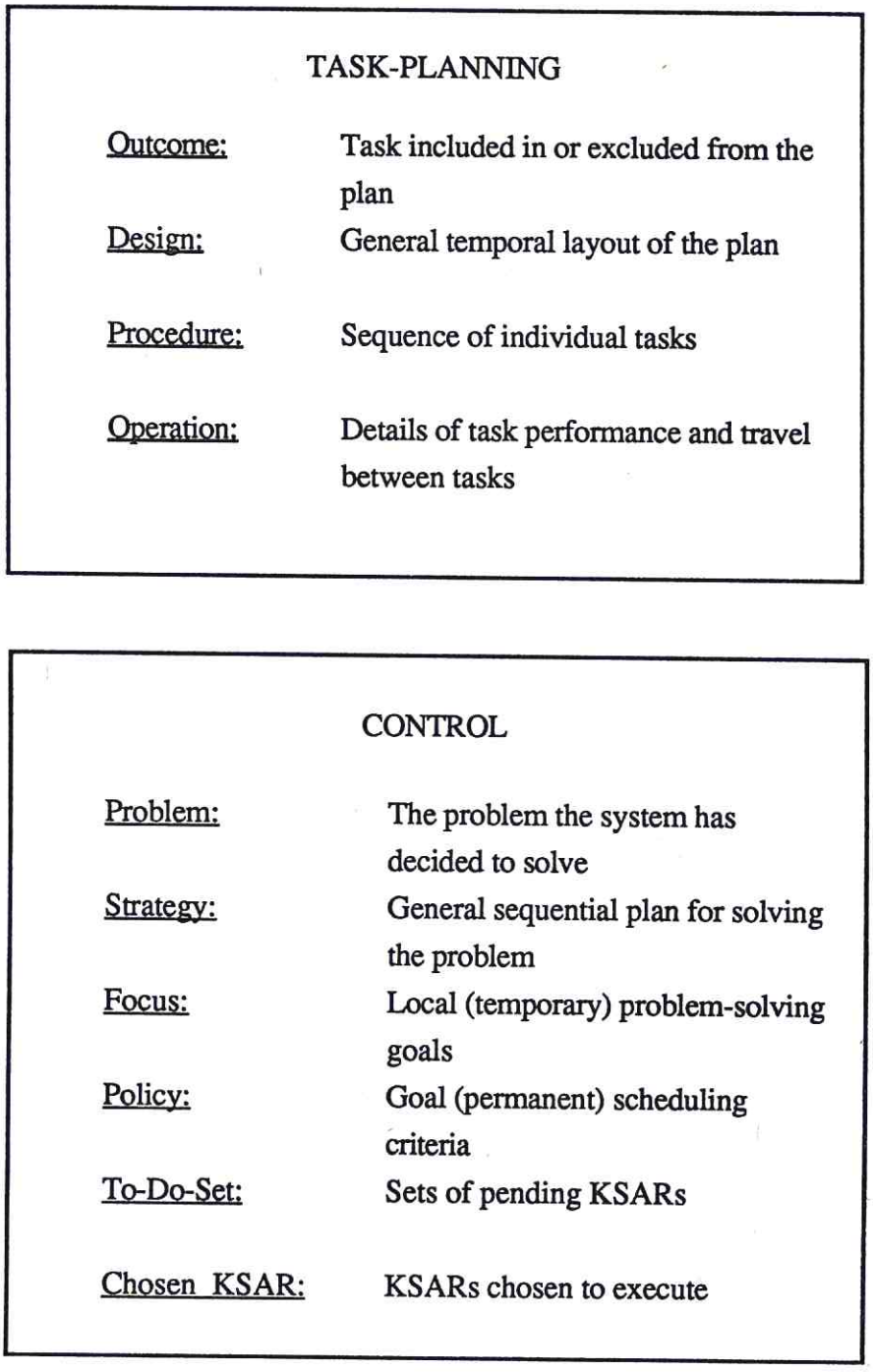
problem solving; the other panel is for expressing control. Both panels have internal structure. The domain panel is a two dimensional structure: its horizontal dimension represents temporal intervals in the plan, the vertical dimension for the OPM planning system represents abstraction levels in the planning process (outcome, design, procedure and operation). The control panel has problem solving cycles as the horizontal dimension (a problem solving cycle corresponds to the selection and running of a knowledge source); the vertical dimension again represents abstraction levels in the control domain (problem, strategy, focus, policy, agenda, scheduled-ksar). The structure of the two panels which comprise a version of the OPM system using the Control Model blackboard are shown in fig. 6 together with their definitions.

In addition to the blackboard, the system has two other structures of primary interest: these are control knowledge sources and the Agenda. The Agenda will be discussed below. Knowledge sources perform the functions which, by now, should be familiar, but, like Hearsay-III, the Control Model requires domain and control KSs. Control KSs respond to events on the control panel of the blackboard and perform control operations. Control operations can refer either to domain Knowledge Sources or to Control KSs: in this way reflexive control is performed. Both kinds of knowledge sources can relate to items on the blackboard (in either panel). As in other systems, KSs are completely independent (and ignorant) of each other: they can communicate only indirectly by modifying the blackboard state.

One interesting observation made in ([Hayes-Roth, 1984]) is that KSs can embody various inference mechanisms. Top-down KSs refine decisions at one level of abstraction into a set of decisions at lower levels of abstraction. Bottom-up KSs abstract a set of decisions at a lower level of abstraction into a more general decision at a higher level of abstraction. Same-level KSs infer decisions at the same level of abstraction as the decisions upon which they operate. Finally, there are inter-panel KSs which make decisions in one panel of the blackboard based upon decisions already taken in other panels of the blackboard.

Because many KSs may be triggered simultaneously,* an intelligent control mechanism is necessary to determine which KS should execute its action part and modify the blackboard on each cycle.

* A KS is triggered when its condition pattern matches an entry on the blackboard.



Abstraction levels for the OPM blackboard
(level names are underlined)

Fig. 6

When a KS is triggered by matching its pattern against some decisions (henceforth: *entries*) on the blackboard, a Knowledge Source Activation Record (*KSAR*) is created (KSARs are directly analogous to activation records in Hearsay-III, but the operations defined over them are different). KSARs record the triggering of some particular KS by a blackboard event or combination of events, together with other information, such as contextual information, KS characteristics, triggering event characteristics and so on. The information provided by KSARs is used by the scheduler, together with its knowledge of pattern characteristics, solution state, problem solving strategy, etc., to decide which KSAR to activate on each cycle. When a KSAR is created, it is put into an agenda. The agenda is the primary data structure of the scheduler.

In a serial implementation, the Blackboard Control Model problem solving cycle is composed of three steps:

1. Update the agenda. The most recently executed KSAR is removed from the agenda. All KSARs created by triggering KSs based upon recent blackboard changes are added to the agenda. Finally, KSARs whose conditions are invalidated by recent changes to the blackboard are removed from the agenda (or given a different status in the agenda);
2. The scheduler selects a KSAR to be executed from the updated agenda;
3. The scheduled KSAR is executed: this typically brings about changes to the blackboard.

In ([Hayes-Roth, 1984], section 3; [Hayes-Roth, 1985], section 3), an alternative to this control mechanism is presented which does away with the 'hard-wired' agenda and scheduler replacing them with KSs in the control panel of the blackboard. The alternative model has three basic knowledge sources: Update-To-Do-Set, Choose-KSAR and Interpret-KSAR. All three KSs operate on the two lowest levels of the control panel. The scheduler, in this model, is unlike many other schedulers mentioned in this paper, because it has no domain-specific knowledge. In order to select a pending KSAR, the scheduler knows only that it must apply some evaluation function to some set of scheduling criteria. Other control KSs which operate at higher levels of the control panel recommend particular evaluation functions and scheduling criteria to apply during particular periods of the problem solving process.

The control model is reflexive. Executed control KSs make recommendations which influence the scheduling of subsequently triggered control KSs as well as the scheduling of domain KSs. The model makes no categorial distinctions between control and domain KSs. The model does not allocate problem solving cycles to control or domain reasoning. Instead KSARs representing all triggered KSs appear on a common agenda: all the KSARs compete for scheduling priority.

These features enable the blackboard control model to reason about the relative priorities of domain and control knowledge sources in exactly the same way in which it reasons about other control issues. The control KSs permit the model to effect an arbitrary number of meta-levels, but without the usual proliferation of architectural levels ([Davis, 1980], [Genesereth, 1983, 1984], [Lenat, 1983]).

The Control Model explicitly represents domain and control knowledge while, at the same time, providing an integrated approach to domain and control problem solving. It can adopt high-level strategies and refine them as focus and policy decisions. This process is conducted incrementally and relies upon the activities of individually triggered, top-down KSs; this feature permits incremental strategy formation in the model. The model can also decide to perform actions, on a more opportunistic basis, which do not fall within the current focus or are just part of the current strategy. The model, furthermore, can induce new high-level strategies from useful non strategic series of focus decisions by the actions of bottom-up control KSs. These new strategies can then be refined in a top-down fashion. The model also requires an extremely simple, adaptive scheduler which has no knowledge whatsoever of the problem domain: this entails that the scheduler is domain-independent and quite general.

The Control Model has been applied to other domains: the HEARSAY-II speech understanding domain and sonar signal understanding as exemplified by HASP ([Feigenbaum, 1982]), amongst others.

Hearsay-II (See section 3.2 above) uses a two-phase strategy. The strategy is implicit, and is buried in the code which implements the scheduler (outlined above, section 3.2). The two-phase strategy is bottom-up as far as the lexical level, followed by an opportunistic phase. Higher level opportunism only occurs after the lower level strategy has produced hypotheses with relatively high confidence.

In addition to the three basic control KSs attended to above, the Control Model version of Hearsay-II uses six control KSs. The two phase control strategy is established by a KS called Problem-

Triggered-Strategy which notices signal interpretation tasks for which hypotheses at an intermediate abstraction level are substantially more reliable than the input data.

The second control KS, called Strategy-Triggered-Tactic is another top-down KS. It implements a multi-phase strategy as a series of tactics. This KS acts upon a level of the control panel called the Tactic Level (this level is an extra one which is added to the standard levels in the Model; it sits between the Strategy and Focus Levels). Tactics in the Hearsay-II model are refined as a series of foci. The KS which performs tactic refinement is identical in structure to the Strategy-Triggered-Tactic, and they could be combined in a more general KS.

The fourth KS is an inter-panel KS which recommends focussing on areas of the domain blackboard in which there are few or unreliable hypotheses. This KS is called Solution-Triggered-Focus.

Problem Type and Current Tactic determine policies. Policy determination is performed by a KS, called Tactic-Triggered-Policies, which operates top-down. This KS is used for both phases of problem-solving, when the system is in the opportunistic phase, this KS prefers KSARs which use efficient KSs, which are triggered on reliable data, which are likely to produce credible hypotheses, which produce hypotheses which can be linked to reliable hypotheses already on the blackboard and will produce hypotheses which substantially improve the overall solution. When the system is in the bottom-up phase, this KS recommends no scheduling policies which leads to the scheduler relying on only focus decisions.

The final KS, again top-down, recommends a deterministic evaluation function for speech-understanding problems. The evaluation function is computed from the weighted sum of ratings of all foci and policies. This KS is called Task-Triggered-Evaluation-Function.

The HASP system, which developed out of the SU programs (see section 3.5) interprets sonar signals in real time. The input signal is analysed by generating hypotheses about different intervals in the signal at different levels of abstraction. The HASP blackboard has as its horizontal dimension time intervals in the signal; the vertical dimension consists of abstraction levels.

HASP generates hypotheses on five abstraction levels. The bottom level is concerned with the sonar signals; the topmost level is the situation board which represents the disposition of signal sources

(ships and submarines). Between these levels, there are three other levels: harmonics within the signal, sources of harmonics such as propellers or engines and vessels.

The control strategy employed by HASP is based upon three event-types. The strategy is nested and iterative. The events are clock events, expected events and simple events. Clock events record the times at which KSs can be invoked. Expected events record events which are expected to occur on the blackboard for a given KS: the KS can respond to the events it expects. Simple events correlate additions or changes to hypotheses with a series of KS which can respond to it.

On each cycle the strategy specifies the following actions:

1. For every due clock event in last-in-first-out order, apply the associated KS to the hypotheses to the associated hypotheses;
2. For every expected event which has been verified, apply the associated KS to the verified hypothesis. Expected events are processed last-in-first-out.
3. Select a simple event and apply all associated KSs to it in the prescribed series. (Again, this is a LIFO queue of events);
4. Repeat stages 2 and 3 until all events have been processed;
5. Repeat stages 1 to 4 until all events have been processed.

The OPM implementation of the HASP strategy uses five control KSs in addition to three basic control KSs.

The first KS establishes the nested, iterative strategy. It operates top-down. The KS specifies no criterion for completing the strategy: HASP has no termination condition because it monitors the signals continuously.

The second KS also operates top-down. It implements the current strategy as a series of Tactics. The third control KS is also top-down. It implements the Tactic as a series of Foci. HASP's scheduling policies are implemented by a KS, called Task-Triggered-Policies, which allocates preferences to KSs. The preferences are used to select KSs to activate.

The final KS in the HASP model operates top-down. It uses the same evaluation function as the HEARSAY-II model.

Of the three systems discussed in ([Hayes-Roth, 1984]; [Hayes-Roth, 1985]), errand-planning, HEARSAY-II and HASP, HASP has the most rigorously defined control scheme and embodies the least opportunitism. The control plan in the model develops in a rigidly top-down fashion, and specifies the ordering of activities down to the level of individual KSARs.

The Control Model is interesting because of its flexibility and for its reflexive properties. The flexibility of the model has been demonstrated in the three domains in which it has been employed: errand planning, speech understanding and signal understanding. The model provides for the explicit representation and intelligent application of control knowledge in a logically separate region of the blackboard. This approach to control (which is similar to the Hearsay-III approach) is different from the control mechanism typically found in AI systems: in the majority of systems, control is embedded in chunks of opaque code.

The model also offers opportunities for a great many control strategies to be employed in one system. Hearsay-II employs two strategies, as was discussed above. It is conceivable that, for complex applications, for example, top-down, bottom-up, best-first and generate-and-test might all be combined in the same system.

However, as Barbara Hayes-Roth states in ([Hayes-Roth, 1984], section 7), despite the evidence supporting the blackboard model of control, the version presented in the paper is only a preliminary formulation which has only been verified by the three experiments described in the paper and outlined above, and is in need of further testing. A major problem which faces the model both as a practical proposition and as a theory of knowledge control is the fact that it introduces extra computational load and increases the amount of storage needed by the system. Despite these negative points, the model is very attractive for the reasons given above.

3.5. SU/X, SU/P and AGE

All of the systems described so far have contained Knowledge Sources (or specialists in the case of the 1979 planning system of the Hayes-Roths) which have their knowledge expressed mainly as procedures. The triggering patterns of the KSs in these systems have been predicates expressed in the language used to implement the blackboard system (usually LISP). The term *pattern-directed* invocation applies to these systems in a fairly loose sense. Also, the internal structure of the KSs in the systems discussed above has tended to be monolithic: there has been a tendency to construct large, multi-functional KSs, expressed as implementation-language routines. OPM and Hearsay-III offer the opportunity to build small KSs, but these are still in LISP.

The systems described in this and the next section take an alternative approach as their basis. The systems described in this and the next section provide facilities for building KSs which have internal structure. The systems do this by building KSs out of sets of production rules ([Davis, 1980] [Waterman, 1978], [Newell, 1973], [Anderson, 1983]).

Before continuing it should be noted that SU/P ([Feigenbaum, 1978]) is the direct precursor of CHRYSALIS. CHRYSALIS adopts an architecture which is slightly different than SU/X as will be described below.

SU/P and SU/X (which later because HASP ([Feigenbaum, 1982])) were developed at Stanford during the late 1970s and early 1980s by Nii, Feigenbaum et al. The task domain of SU/P is the interpretation of protein X-ray crystallographic data; that of SU/X is sonar signal interpretation.

Both of the systems are partly concerned with performance issues. SU/P has to cope with an immense search space in which, not only is there comparatively little to learn from human experts, but also the effective choice of operations at an early stage in the problem solving process critically impacts upon the success of the derivation of adequate solutions. SU/X, on the other hand, has to perform in real time; ships and submarines move, and, in a conflict situation, it is imperative to have as comprehensive an analysis as possible - this entails real-time performance.

The same basic design philosophy was followed in both SU/X and SU/P. They also have architectural features in common: both systems are designed to handle large volumes of digitised physical

signals. Both systems inherit features from the HEARSAY-II system, but, as has been hinted above, differ in some respects. The main legacy from HEARSAY-II is a multi-level database, which is still called the blackboard. The two systems differ from HEARSAY-II in the following respects:

(i) knowledge is represented as production rules;

(ii) control structure is represented as knowledge sources - knowledge sources encode problem solving methods and strategies;

(iii) both programs can explain their reasoning by tracing the hypotheses on the blackboard, and

(iv) the design is deliberately made as general as possible to facilitate application to other domains.

In SU/X and SU/P, knowledge source function is extended so that, not only can they make inferences, but they can also check the inferences made by other KSs. As in the other systems so far discussed, hypotheses on the blackboard are linked together to form networks. Hypothesis networks can span levels on the blackboard (the blackboard is a hierarchical structure). The hypothesis network is rooted at the lowest level of the blackboard (where the physical signals are received) and are integrated with elements in the highest level. The highest level constitutes a description of the hypothesised problem solution. The network is termed the current best hypothesis.

Within this general framework, the KSs contain a considerable body of domain knowledge, both factual and heuristic. Knowledge sources also represent control knowledge, a fact which establishes a commonality between these systems, OPM, Hearsay-III and the Hayes-Roth's 1979 planning system. A further characteristic of SU/X and SU/P which is common with other systems is the opportunistic process of hypothesis formation. However, the opportunistic paradigm (which, like HEARSAY-II is a hypothesise-and-test process) is conducted in terms of data-driven and model-driven hypothesis formation techniques. One task which is given to the control component is the determination of the applicable method given current circumstances.

In SU/X and SU/P, the unit of processing activity is the event. Events are divided into three types: knowledge-based events, which are related to changes in the hypothesis; time-based events, which are related to expectations of what will happen at some point in time; problems, which are also expecta-

tions based upon the program's model of the situation for which evidence either way has not been found.

The terms data-driven and model-driven need some explanation before moving on to a more detailed description of the SU programs. By data-driven, Nii and Feigenbaum intended that results are inferred from the input data. By model driven they intend inferences which are based upon expectations inferred from domain-specific knowledge. The choice between the strategies in signal-processing domains is not determined by hard-and-fast criteria, but, rather, from the KSs which are available at any one time or in any particular implementation and upon the strength of the model of the domain. The combination of the two strategies exemplifies the need for multiple knowledge sources which apply to the same aspect of a given problem ([Stefik, 1982]).

In the SU systems, the blackboard is divided into two planes: the hypothesis plane and the control plane. Both planes are hierarchically organised. In both SU/X and SU/P, control is organised in a hierarchy. The highest level of the hierarchy is concerned with strategies. The middle level is the KS-Activation level. At the bottom of the hierarchy, there is the Hypothesis formation level. The Hypothesis formation level contains KSs which are to make primary inferences in the domain-specific plane. The intermediate level contains KSs which represent meta-knowledge: these KSs comprise the KS-Activation level. KSs on the KS-Activation level know about the capabilities of the KSs on the Hypothesis formation level. KSs at the Activation level represent policies about knowledge utilisation. At the highest level of the control hierarchy is the strategy level: there is only one KS at this level. The single KS analyses the current solution and determines which part of the blackboard to analyse next and it determines what the next strategy to be employed will be.

The control hierarchy in the SU systems is not at all coupled to the domain hierarchy. The domain hierarchy represents an *a priori* plan for the solution (the same point applies to all hierarchically organised blackboards). The control hierarchy (or plane) represents an organisation of the activities necessary to solve any particular instance of the generic problem.

The blackboard structure of the SU programs is somewhat different than in other systems. The differences (adopted, it would appear, by AGE and CHRYSALIS) are worthy of some attention.

The first point to make about the blackboard in the SU programs is that Nii and Feigenbaum in ([Feigenbaum, 1978]) regard it as a control structure and not just a passive database. This alternative view of the blackboard has important implications for its implementation.

The blackboard for the SU programs consists of three lists and two global variables, as compared with some complex indexing scheme. The blackboard consists, in detail, of the following structures:

- The current best hypothesis: this is formed by the application of domain and control knowledge;
- The event list. The event list records changes in the hypothesis which have not yet been processed by any KS. An event also contains the name of the KS and the identifier of the rule which caused that change. (Note, that this point, which is taken almost verbatim from ([Feigenbaum, 1978]), gives a strong hint that KSs in SU/X and SU/P are collections of productions) - this point is not made explicitly in Feigenbaum's chapter.
- The event. This is a global variable which contains the currently 'active' event. The currently active event is an event currently being processed by some KS. The event also represents in a very simple and elegant form the current focus of attention.
- The problems-list. This is a list of unresolved problems which have been encountered by various KSs. Unresolved problems range from expected data which is not yet available to detectable errors in the program (such as insufficient knowledge).
- The event history list. The event in conjunction with its predecessor and successor events form a causal chain of reasoning: they also permit a limited amount of scope for temporal reasoning. In signal processing and analysis, the event history list can be used by KSs to analyse a series of events which have occurred over a period of time. In addition, it can be used for the generation of explanations.

The general features of the SU programs can also be discerned in the AGE system ([Nii, 1979]).

AGE stands for Attempt to GEneralise. It is a Knowledge-based program for building knowledge-based programs: its shell-like properties are similar to those of Hearsay-III, but it also contains tools and strategies to assist knowledge engineers in building blackboard-based systems.

In their paper on AGE (Nii, op.cit), Nii and Aiello state that their goal is to "demystify and make explicit the art of knowledge engineering". To this end, the AGE system is designed as an extensible set of tools and strategies for building systems based on the blackboard model. The interface provided by AGE is of little concern to the current discussion which is centred on the architecture of the system and its library. The AGE blackboard framework is shown schematically in fig. 7.

The version of AGE described in (Nii, op.cit) contains a blackboard, facilities for building Knowledge Sources and precompiled control facilities which belong to a control library. In AGE, KSs respond to changes on the blackboard.

The AGE blackboard is, again, hierarchically organised in a fashion reminiscent of the SU programs. AGE permits strict or loose hierarchies, and can support an arbitrary number of planes or panels. The hypothesis formation model adopted by AGE is based upon Hempel's characterisation of hypothesis formation ([Hempel, 1966]).

An hypothesis can be generated

- deductively using support from above - this support can be (a) theoretical support (or model-based support), (b) support from more inclusive hypothesis that have independent evidential support, or
- inductively, using evidential support from below. (Nii, op.cit. p.546).

Hypotheses are created by KSs. In AGE, the representation of KSs is based on the assumption that there are at least four broad categories of knowledge needed to solve problems. The four categories are:

1 Knowledge of specifics;

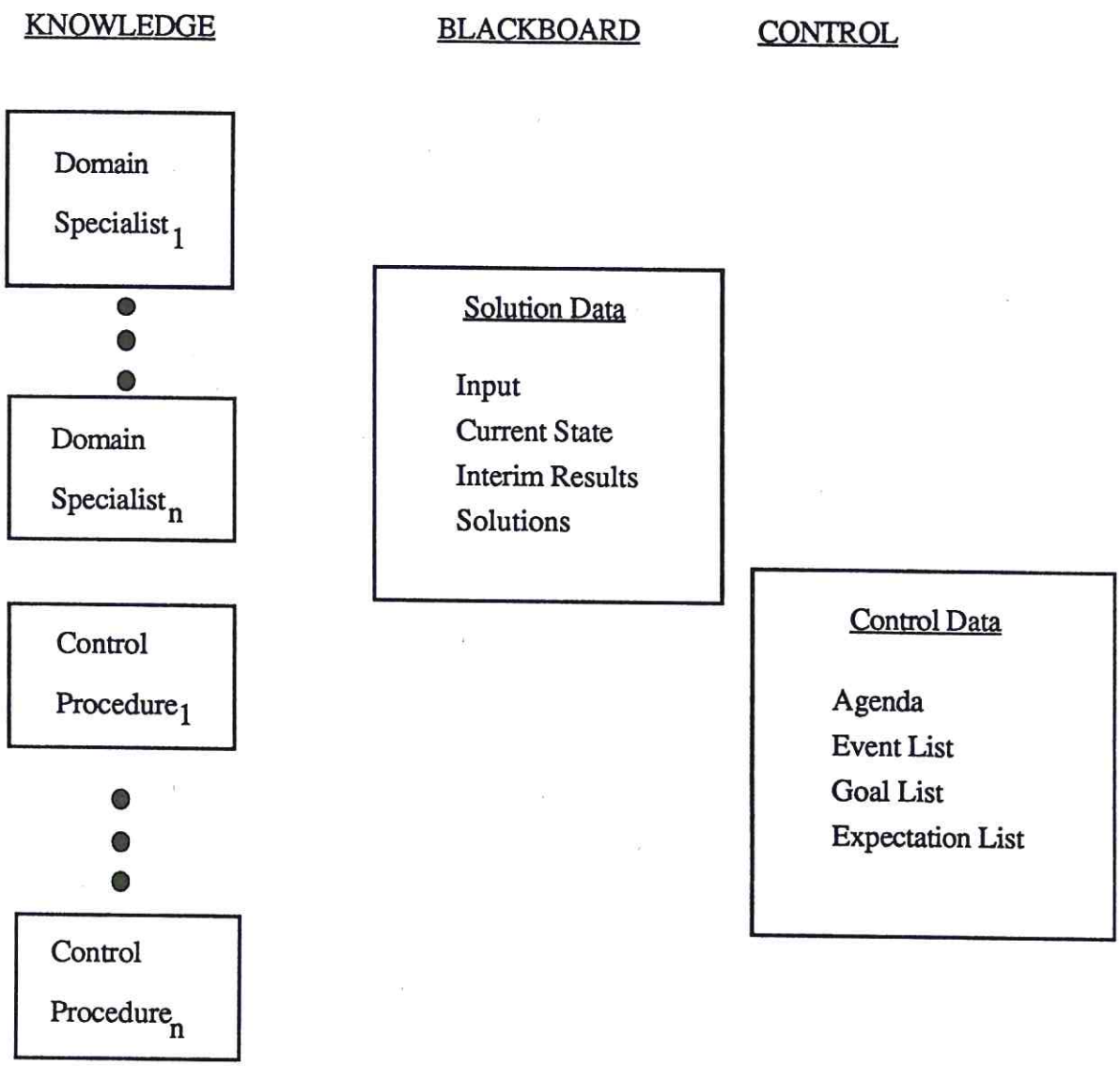
2 Knowledge of how to deal with the specifics;

3 Knowledge of universals and abstractions in the task domain;

4 Knowledge of problem solving and knowledge utilisation methods in the domain.

This classification is important to the tool-kit aspects of AGE and also informs the representation of knowledge in the system.

Knowledge Sources in AGE have the standard functions which are found in all the systems described in this paper. KSs in AGE can be model- or event-driven. In the model-driven scheme, KSs



AGE - schematic
(from Nii, talk, June, 1985)

Fig. 7

are organised around conceptual entities: the conceptual entities represent models. The entities may be unique within a model, or may be arranged in clusters. In the model-driven scheme, KSs operate very much like schemata or frames. In the event-driven scheme, KSs are organised around events: they are clustered around system events such as input or the creation of a new entity. This is another distinction which makes AGE an interesting system: its representational basis makes quite clear the relationship between static and dynamic concepts.

Knowledge Sources in AGE, as in the SU programs and CHRYSALIS, are constructed from production rules. KSs serve to collect the production rules in the system according to some logical relationship. Each KSs has five components:

(a) a list of events which act as preconditions for the activation of the KS;

(b) a list of hypothesis levels on the blackboard which the KS spans;

(c) a list of links generated by the KS (note that these are determined on a static and a priori basis);

(d) a strategy which is to be used by the rule (either single or multiple hit);

(e) a facility for variable bindings to set local context, to avoid multiple evaluations of the same expression (note: this tells us that KSs are subject to an instantiation process, which is not explicitly stated), and to permit communication between condition and action parts of rules.

Rules within KSs also have credibility values or weight: these reflect the reliability of the rule in a manner akin to certainty factors in MYCIN ([Shortliffe, 1976]) and PROSPECTOR ([Duda, 1978]).

The control component in AGE can be user-defined, as in Hearsay-III, but more features are preprogrammed into the system. The system requires a control kernel. The functions of the control kernel are as follows, (Nii, op. cit, p.649).

When a KS has been selected for invocation, each rule it contains is evaluated. The condition-part or LHS (Left-Hand Side) of each rule is evaluated in the context of the current blackboard state. When the LHS meets the evaluation criteria (i.e., evaluates to true) the rule is fired. Next, the rules which have fired can generate inferences. Inference generation can be changing hypotheses, indicating that some event is expected, or indicating that some hypothesis needs to achieve some value.

The third part of the control component is concerned with focus of attention. This mechanism is concerned with the section of KSs which are relevant to the task in hand.

The operations of the kernel are fairly complex, so AGE provides a set of control macros to simplify the construction of kernels. The macros provided in the system are concerned with event- and model-driven strategies.

The event-driven strategy examines entities at a lower level of the blackboard and uses them as evidence for higher-level hypotheses. AGE permits events to be processed by rules on the basis of first-in-first-out, first-in-last-out or best-first: these roughly implement breadth, depth and best-first strategies. When an element is chosen, it becomes the focus of attention. The event-driven strategy is implemented in a manner similar to that in the SU programs.

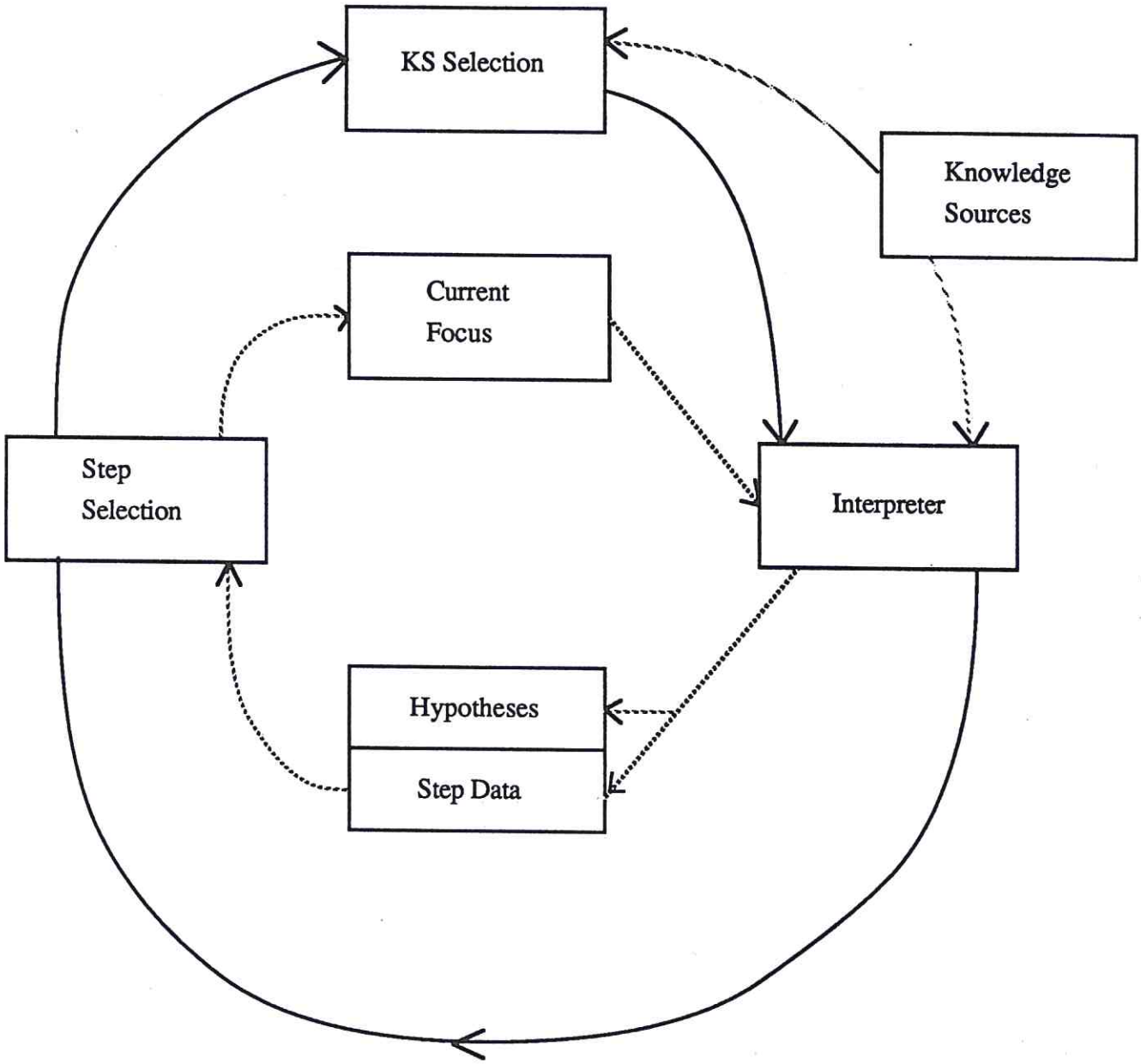
The expectation-driven macro helps the construction of expectation-driven strategies. As was stated above, expectation-driven strategies are clustered around objects and their attributes. The AGE control model is shown in fig. 8 (from Nii, op. cit.).

Appendix II of the Nii and Aiello paper contains details of the control kernel.

The AGE system differs from many of the other systems in this paper in its development of the epistemological basis of a variant of the blackboard model and in its structuring of the blackboard, which was derived from the SU programs. It also provides the fairly general mechanisms for controlling applications, although the standard facilities are not open to inspection by KSs, as is the case with OPM and Hearsay-III.

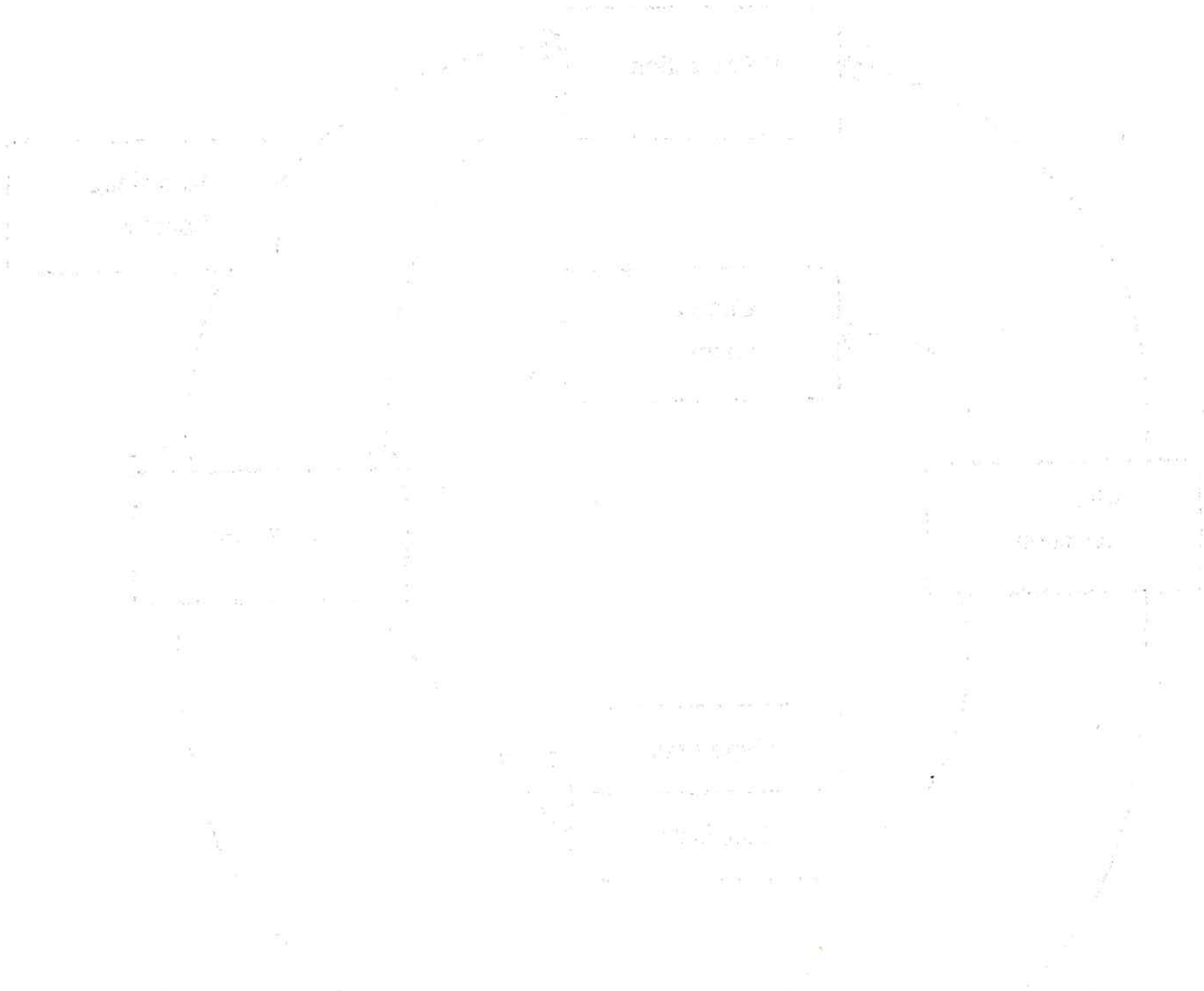
AGE, despite its basis in signal-processing systems, has been successfully applied to other domains. It has also been used to re-implement PUFF, an expert system for pulmonary function analysis ([Aiello, 1983]), ([Kunz, 1978]). PUFF has also been re-implemented as CENTAUR ([Aikins, 1980],[Aikins, 1983]), a frame-based production system which introduces contexts and prototypical knowledge into the problem solving process.

The re-implementation of PUFF in AGE is really a case study of three control strategies: it constitutes three implementations, not one. It appears that, given the user-friendly nature of AGE, it did not



AGE Control Model

Fig. 8



take very long to perform the implementations. The three strategies were event-driven, model-driven (both familiar from AGE) and backchaining (a new strategy for AGE, and well-known from MYCIN ([Shortliffe, 1976])). The point of the re-implementation task was to determine which of the three control strategies is the most appropriate for PUFF's task. The system, called AGEPUFF, used the same domain knowledge (five Knowledge Sources), but varies the control strategies which control the domain knowledge. The fact that the same domain knowledge can be used in different control regimes is a definite plus both for AGE and for the blackboard model. The variation in control structure entails that the configuration of the resulting blackboard structures will be different for each control strategy: that is the concern of the control strategy and not the domain knowledge or the conflict resolution strategy (as it is in OPS-5, [Forgy, 1977]).

For the curious, all the strategies employed in the study were divide-and conquer approaches, but they differ in the way in which domain knowledge is divided. Of the three strategies, the expectation driven strategy won out: it represents a compromise between data- and (implicit) goal-driven strategies. Given a good model of the solution space, an efficient system could be built using the expectation-driven model. One fault with the strategy is that it is possible for some solutions to be missed.

The VM ventilator management system ([Fagan, 1980]) was also re-implemented in AGE: this, too used an expectation-driven strategy ([Aiello, 1983]).

The point about mentioning these re-implementations is to show that AGE and the blackboard model are flexible and powerful enough to re-implement systems which were originally constructed using other, very different architectures (PUFF was originally implemented in EMYCIN ([van Melle, 1979])). What is not stated in the Aiello paper (Aiello, op.cit.) is the comparative performance of each implementation: this would be most illuminating.

3.6. CHRYSALIS

CHRYSALIS ([Terry, 1983]), like HASP/SIAP ([Feigenbauum, 1982]) is a descendant of the SU programs, SU/X and SU/P, described in the last section. HASP developed from SU/P, and CHRYSALLIS grew out of SU/X.

Just as the SU programs and AGE differ from many other blackboard systems in their use of production rules in KSs, CHRYSALIS differs from the other systems in its structure. CHRYSALIS has a blackboard as its primary data structure, and its Knowledge Sources are built from production systems. However, CHRYSALIS is structured as a Hierarchical Production System. Hierarchical Production Systems introduce extra characteristics into the blackboard architecture. The hierarchical production system was selected because of the way in which it handles the focus of attention problem.

The task domain of CHRYSALIS is the same as SU/X: X-ray protein crystallography. A major problem which CHRYSALIS faces is that, in addition to the large search space, there are a great many heuristics which can be applied and the widening of the application of the heuristics is then important. Also, as Terry remarks (Terry, op.cit., p.45)

> "The system's resources are also far too limited (the solution of a small protein currently consumes most of the address space and about ten hours of CPU time on the SUMEX dual PDP KI-10, even with compiled code) and the knowledge is far too weak to permit either exhaustive invocation of rules or leisurely exploration and backtrack. If the system is to find the correct answer it must somehow pick a nearly optimal sequence of rules to apply in response to the specific problem. This is especially true for a production system that does not exhaustively apply the rules in the conflict set. If some rules are not fired when they match, some solutions may be lost. Proper focus is then critical."

Even though CHRYSALIS differs in many repects from a conventional blackboard system, it still has a blackboard as its main data structure. The blackboard is divided into two panels: one for hypotheses, the other for representing electron density maps. For a detailed discussion of the blackboard and its relationship to the domain, readers should consult (Terry op. cit): there is insufficient space here to give a detailed description. Suffice it to say that each plane is divided into a hierarchy of abstraction levels. CHRYSALIS inherits some of the features of its blackboard from SU/X, but does not have a plane dedicated to control issues. Instead, CHRYSALIS relies upon the hierarchical production system to perform control functions: control is expressed as Knowledge Sources which are constructed from production rules (as in SU/X, SU/P and AGE). Within CHRYSALIS, the production rules which form the KSs

communicate via the blackboard, so the system is properly considered to be an example of the blackboard model. The abstraction levels in the two panels of CHRYSALIS are depicted in fig. 9.

Since CHRYSALIS is somewhat different from other blackboard systems, it is worthwhile spending some time describing the architecture of the system in some detail.

The hierarchical production system (HPS) employed in CHRYSALIS has a complete production system in each level of the hierarchy. In CHRYSALIS, the productions at each level examine the current situation and select actions to perform at the next lower level. The bottom most level (the largest) consists of the object-level rules which construct the solution. The higher levels are composed of control rules which invoke KSs instead of performing object-level computations.

Control within the HPS is top-down. The highest level, as in the SU programs, contains a single control KS. The conceptual unit of control is the KS (a set of production rules) rather than the individual rule. The highest level KS examines the datbase and selects a KS or a sequence of KSs to execute at the next lower level. This process is repeated until the object-level rules are invoked. When one KS invokes another, control passes to the invoked KS: in other words, there is a procedure-call relationship between invoking and invoked KSs, and control returns to the invoking KS only when the invoked KS has run to completion or has run out of rules to fire.

An HPS, it is argued by Terry, has some advantages over conventional production systems. The first reason is clarity. Traditional production systems require extra context-sensitive elements in their conditions by which control information can be communicated (via 'signals' or items placed in working memory to force certain productions in the production set to be considered). Since rules in the HPS are collected into Knowledge Sources, the productions they contain need not contain such context-sensitive condition elements: that is handled by the context conditions which determine KS selection.

Representation of control heuristics as rules contributes to the clarity as they are written in a highly constrained representation and tend to be quite small. The same arguments are applied to meta-rules by Davis ([Davis, 1980]).

The second benefit cited by Terry (p.47) is that the HPS employs a direct control mechanism. If there are specific strategies about what to do and when to do it, there ought to be no good reason to

wait for actions to come to the top of the agenda for the right goal to appear (as is the case with OPM). CHRYSALIS' HPS permits control rules to fire a KS, which imparts directness of control (as was noted above in connection with the procedure-call relationship between KSs).

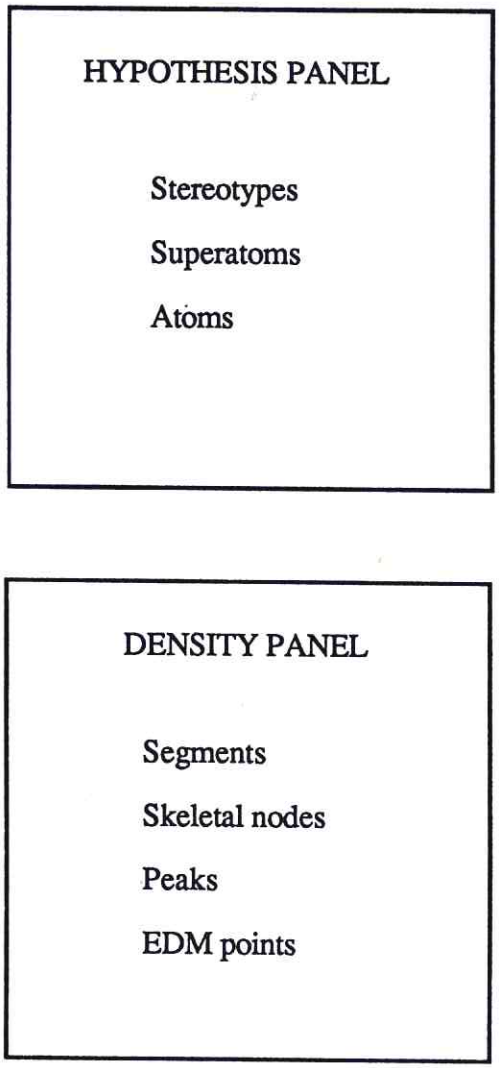
Within CHRYSALIS, control works top-down. Strategy rules do not examine the hypothesis directly, but, instead, inspect a summary of the blackboard, the FEATURESLIST, which is computed each time control returns to the strategy level (the FEATURESLIST represent parts of the amino acid sequence in the protein). The FEATURESLIST is computed by examining the current best hypothesis (cf., the SU programs). The action of a strategy rule is to invoke a task KS or series of task KSs in response to some pattern it sees in the developing hypothesis.

The middle level of the hierarchy is the task level. Each KS at this level (called a task) is a rule set which directs the activation of object-level KSs to solve a sub-problem identified by the task. Each task has its own area of expertise and a region of the data/solution space (the latter is defined by the pattern of features matched against that region by the invoking KS): these characteristics together define a focus for the task.

As in HEARSAY-II, task rules inspect the blackboard, but only react to changes. CHRYSALIS optimises the task of blackboard change detection by packaging changes into events (cf. SU/X). The changes in the blackboard constitute an event-driven control mechanism. Event processing, as in AGE, can operate breadth-, depth- or best-first, depending upon where events are added to the event list.

The bottom level consists of KSs which perform genuine domain-related problem solving actions. KSs at this level are not restricted, but, instead, can examine the entire blackboard. KSs at this level can change data or hypotheses. The bindings for KSs at this level (all KSs have bindings as in AGE) signal that the KS has an action which is cyclic, single or multiple hit. KSs at this level ought to be complex, but quite a lot of the complexity has been delegated to the many access functions which interface the productions to the database and FORTRAN number-crunching routines.

The main point about CHRYSALIS is that it employs a different control structure. The conventional blackboard architecture is augmented by a control structure reminiscent of conventional programming languages. CHRYSALIS clearly has a lot in common with the SU programs, but employs meta-



CHRYSALIS blackboard hypothesis and data panels
(from Terry, 1983)

Fig. 9

level reasoning in an effective way to control the inferences possible in the system: in this respect, CHRYSALIS represents a development of meta-rules.

3.7. Ariadne-1

Ariadne-1 ([Craig, 1986]) is a derivative of the OPM type of blackboard. Ariadne-1 is a flexible shell to be used in the construction of blackboard systems.

Ariadne-1 is a multiple-panel system and uses an agenda-based scheduler. As with OPM and OPM's successor, BB-1 ([Hayes-Roth, 1984b]), it is possible to represent control information explicitly in a separate panel on the Ariadne-1 blackboard and to use the system-provided agenda to schedule control KSs and domain KSs.

Ariadne-1's architecture is somewhat more flexible than OPM or BB-1. Ariadne-1 permits applications to create panels and levels dynamically. This is made possible by the fact that the data structures in Ariadne-1 which represent panels and levels are available for inspection and modification by the user. The thinking behind the dynamic creation of panels and levels is that in poorly understood domains, the initial problem decomposition may not be correct and that KSs could be constructed to recognise this fact and introduce extra architectural partitions. The background to Ariadne-1 is made clear by these comments: Ariadne-1 was designed as a shell for learning systems.

Ariadne-1 also permits the user to use meta-operators. Meta-operators in Ariadne-1 can inspect KSs as well as entries in a level. Meta-operators are also provided for the dynamic panel and level creation processes described above.

The final way in which Ariadne-1 is slightly different than the other systems described in this chapter is in the structure of blackboard entries. Entries in Ariadne-1 have predefined slots which contain linkage and history information. The latter slots record the time of creation and the identity of the KS which created the entry. The creation slot also records the identifiers of the entries which caused the KS to execute and, thereby, create the entry. Other history information is recorded: the modifications to an entry are recorded in the entry. Modification data contains the identifier of the modifying KS and details of the modification. The history slots are present so that KSs can reason about the derivations of entries -- again,

there is a strong bias towards learning in the existence of these slots.

4. SUMMARY

This paper has described some of the more important blackboard systems. It began with an outline of the blackboard model of problem solving and its metaphor. From the model, a number of systems have been constructed: the set of features they have in common is referred to as the blackboard architecture. As Hayes-Roth notes ([Hayes-Roth, 1983]), the blackboard architecture is an informal construct. The informality of the architecture explains why various implementations have concentrated on different aspects of the model.

The architecture can be summarised as follows. A system can be described as a blackboard if it has the following properties:

- A central, hierarchically organised database called the *blackborad*. The blackboard is basically organised as an abstraction hierarchy, although other structural partitions (panels or planes) are permitted.
- A set of *Knowledge Sources*. Knowledge Sources represent the problem solving knowledge present in a blackboard system. Knowledge Sources are permitted to communicate with each other only via placing and modifying items on the blackboard. Knowledge Sources may not communicate in any way other than via the blackboard.
- A centralised scheduler. The scheduler is responsible for controlling the execution of Knowledge Sources in such a way that solutions are generated in a coherent manner. The scheduler executes Knowledge Sources according to one or more problem-solving strategies.

The blackboard architecture is roughly defined by these requirements. All of the systems reviewed in this paper can be seen to have these properties. A more detailed definition of the architecture, together with a discussion of the role of opportunism in blackboard systems can be found in ([Craig, 1987b]).

REFERENCES

[Aiello, 1983] Aiello, N A comparative study of control strategies for expert systems: AGE implementation of three variations of PUFF. Proceedings of the National Conference on Artificial Intelligence (AAAI-83), 1 - 4, 1983

[Aikens, 1980] Aikens,J.S. Protypes and Production Rules: A Knowledge Representation for Computer Consultations. Rept. STAN-CS-80-814, Stanford University, 1980

[Aikens, 1983] Aikens, J.S. Prototypical Knowledge for Expert Systems. AIJ, Vol. 20, No. 2, pp. 163 - 210, Feb. 1983

[Anderson, 1983] Anderson, J.R., The Architecture of Cognition. Harvard University Press, London, 1983

[Balzer, 1980] Balzer, R., Erman, L., London, P. and Williams, C. HEARSAY-III: A Domain Independent Framework for Expert Systems. Proc. First Annual Conference on Artificial Intelligence, pp. 108 - 110, 1980

[Clocksin, 1982] Clocksin, W and Mellish, C, Programming in Prolog. Springer-Verlag, Berlin, 1981

[Colmerauer, 1975] Colmerauer, A. Les grammaires de metamorphose. Groupe d'Intelligence Artificielle, Marseille-Luminy, 1975

[Craig, 1986] Craig, I.D. The Ariadne-1 Blackboard System. Computer Journal, Vol. 29, No. 3, pp. 235 - 240, 1986

[Craig, 1987] Craig, I.D. BB-SR. Technical Report No. 94, Department of Computer Science, University of Warwick, February, 1987

[Craig, 1987b] Craig, I.D. The Blackboard Architecture: A Definition and Its Implications. Technical Report No. 98, Department of Computer Science, University of Warwick, March, 1987

[Davis, 1980] Davis, R. Meta-rules: Reasoning about control. AIJ, Vol. 15, pp. 179 - 222.

[Doyle, 1979] Doyle,J., A Truth Maintenance System. AIJ, Vol. 12, pp 231 - 272, 1979

[Duda, 1978] Duda, R.O., and Hart, P.E., Nilsson, N.J., and Southerland, G.L. Semantic Network Representation in Rule-based Inference Systems. In [Waterman, 1978], pp. 203 - 222.

[Erman, 1975] Erman, L.D. and Lesser, V.R., A Multi-level Organisation for Problem Solving Using Many, Diverse, Cooperating Sources of Knowledge. Proc. IJCAI 4, Vol. 2, pp. 483 - 490, 1975

[Erman, 1981] Erman, L.D., London, P. and Fickas, S. The Design and an Example Use of HEARSAY-III. Proc. IJCAI 7, Vol. 1, pp. 409 - 415, 1981

[Fagan, 1980] Fagan, L.M. VM: Representing time-dependent relations in a medical setting. Doctoral Dissertation, Heuristic Programming Project, Dept. of Computer Science Stanford University, 1980

[Feigenbaum, 1978] Feigenbaum, E. and Nii, H.P., Rule-based Understanding of Signals. Pattern -Directed Inference Systems, D.A. Waterman and F. Hayes-Roth (Eds.), Academic Press, 1978

[Feigenbaum, 1982] Feigenbaum, E., Nii, H.P., Anton, J.J. and Rockmore, A.J. Signal-to-signal transformation: HASP/SIAP case study. AI Magazine, Vol. 3, pp 23 - 35, 1982

[Forgy, 1977] Forgy, C.L. and McDermott, J. OPS, A Domain-independent Production System Language. Proc. IJCAI 5, pp 933 - 939

[Genesereth, 1984] Genesereth, M.R. Partial Programs. Memo HPP-84-1, Heuristic Programming Project, Dept. of Computer Science, Stanford University, 1984

[Genesereth, 1983] Genesereth, M.R. The MRS Casebook. Memo HPP-83-26, Heuristic Programming Project, Dept. of Computer Science, Stanford University, 1983

[Goldman, 1978] Goldman, N. AP3 User's Guide. USC/ISI, Marina del Rey, CA, 1978

[Hayes-Roth, 1977] Hayes-Roth, F. and Lesser, V.R., Focus of Attention in the Hearsay-II speech understanding system. Proc. IJCAI 5, pp. 27 - 35, 1977

[Hayes-Roth, 1979] Hayes-Roth, B. and Hayes-Roth, F., A Cognitive Model of Planning. Cognitive Science, Vol. 3., pp. 275 - 310, 1979

[Hayes-Roth, 1983] Hayes-Roth, B. The Blackboard Architecture: A General Framework for Problem Solving? Report No. HPP-83-30, Heuristic Programming Project, Computer Science Dept., Stanford University, Palo Alto, CA, May 1983

[Hayes-Roth, 1984] Hayes-Roth, B. A Blackboard Model of Control. Report No. HPP-83-38, Heuristic Programming Project, Computer Science Department, Stanford University, Palo Alto, CA., June 1983 (Revised December, 1984)

[Hayes-Roth, 1984b] Hayes-Roth, B. BB1: An Architecture for blackboard systems that control, explain, and learn about their own behavior. Technical Report HPP-84-16, Stanford University, 1984

[Hayes-Roth, 1985a] Hayes-Roth, B and Hewett, M. Learning Control Heuristics in BB-1. Technical Report HPP-85-2, Stanford University, 1985

[Hayes-Roth, 1985b] Hayes-Roth, B. A Blackboard Model for Control. Artificial Intelligence, Vol. 26, pp. 251 - 322, 1985

[Hayes-Roth, 1986] Hayes-Roth, B., Garvey, A., Johnson, M.V. and Hewett, M. BB*: A Layered Environment for Reasoning about Action. Technical Report No. KSL 86-38, Knowledge Systems Laboratory, Stanford University, 1986

[Hempel, 1966] Hempel, C.G. Philosphy of Natural Science. Foundations of Philosophy Series, Prentice-Hall, NJ, 1966

[Hewitt, 1972] Hewitt, C. Description and Theoretical Analysis (using schemata) of PLANNER. MIT AI Lab., TR-258, 1972

[Kowalski, 1979] Kowalski, R. Logic for Problem Solving. North Holland, Oxford, 1979

[Kunz, 1978] Kunz,J., Fallet, R., McClung, D., Osborn, J., Votteri, B., Nii, H.P., Aikens, J.S., Fagan, L. and Feigenbaum, E. A physiological rule based system for interpreting pulmonary function test results. HPP-78-19 (Working Paper), Heuristic Programming Project, Dept. of Computer Science, Stanford University, 1978

[Lenat, 1983] Lenat, D.B., Threory Formation by Heuristic Search.The Nature of Heuristics II: Background and Examples. AIJ, Vol. 21, Nos. 1,2, pp. 31 - 60, March, 1983

[McDermott, 1974] McDermott, D. and Sussman, G.J. The CONNIVER Reference Manual. MIT AI Laboratory Memo 259a, 1974

[Mycroft, 1984] Mycroft, A. and O'Keefe, R.A. A Polymorphic Type System for Prolog. AIJ, Vol. 23, No. 3, pp 295 - 308, August, 1984

[Newell, 1973] Newell, A. Production systems: models of control structures. In W.G. Chase (ed.), Visual Information Processing, Academic Press, New York, 1973

[Nii, 1979] Nii, H.P. and Aiello, N. AGE: A knowledge-based program for buidling knowledge-based programs. Proc. IJCAI 6, pp. 645 - 655

[Nii, 1986a] Nii, H.P. The Blackboard Model of Problem Solving. Artificial Intelligence Magazine, Vol. 7, No. 2, pp. 38 - 53, 1986

[Nii, 1986b] Nii, H.P. Blackboard Systems Part Two: Blackboard Application Systems. Artificial Intelligence Magazine, Vol. 7, No. 3, pp. 82 - 106, 1986

[Sacerdoti, 1974] Sacerdoti, E.D. Planning in a Hierarchy of Abstraction Spaces. AIJ, Vol. 5, pp. 115 -135, 1974

[Sacerdoti, 1977] Sacerdoti, E.D. A Structure For Plans and Behavior. Elsevier, New York, 1977

[Schank, 1982] Schank, R.C., Dynamic Memory. Cambridge University Press, London, 1982

[Shortliffe, 1976] Shortliffe, E. Computer-Based Medical Consultations: MYCIN. Elsevier, New York, 1976

[Stefik, 1982] Stefik, M., Aikens, J., Balzer, B., Benoit, J., Birnbaum, L., Hayes-Roth, F and Sacerdoti,E. The Organization of Expert Systems: A Prescriptive Tutorial. Xerox, PARC, VLSI-82-1, 1982

[Sussman, 1977] Stallman, R.M and Sussman, G.J. Forward Reasoning and Dependency Directed Backtracking. AIJ, Vol. 9, pp 135 - 196, 1977

[Terry, 1983] Terry, A. The CHRYSALIS Project: Hierarchical Control of Production Systems. Memo HPP-83-19, Heuristic Programming Project, Computer Science Dept., Stanford University, Palo Alto, CA., May, 1983

[van Melle, 1979] van Melle, W. EMYCIN: A domain-independent production-rule system for consultation programs. Doctoral dissertation, Heuristic Programming Project, Dept. of Computer Science, Stanford University, 1980

[Warren, 1977] Warren, D.H.D. Implementing Prolog. DAI Research Report No. 39, Edinburgh University, 1977

[Waterman, 1978] Waterman, D.A. and Hayes-Roth, F. (eds) Pattern-directed inference systems. Academic Press, New York, 1978

[Wilensky, 1983] Wilensky, R. Planning and Understanding. Addison-Wesley, London, 1983